

AD-A281 528



WL-TR-94-1049



**DETAILED EXAMPLE OF USING PIWG
ON THE SUN WORKSTATION AND THE
DEC VAX COMPUTER**

Major Steven A. Davidson
Software Concepts Group
Avionics Logistics Branch
Systems Avionics Division

December 1993

Final Report for Period 6 June 1993 to 18 June 1993

Approved for Public Release; Distribution Unlimited

94-21743 *copy*
 [scribble]

DTIC QUALITY INSPECTED 5

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-7409

DTIC
ELECTE
JUL 14 1994
S B D

94 7 13 019

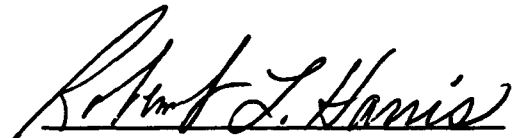
NOTICE


When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility nor any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication, or otherwise in any manner, construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


STEVEN A. DAVIDSON, Major
U.S. Air Force Reserves


ROBERT L. HARRIS, Section Chief
Software Concepts Group
Avionics Logistics Branch


OMER KEENER, Acting Chief
Avionics Logistics Branch
Systems Avionics Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify WL/AAAF, Wright-Patterson AFB, OH 45433-7409 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE December 1993	3. REPORT TYPE AND DATES COVERED Final, 6Jun93 through 18Jun93		
4. TITLE AND SUBTITLE Detailed Example of Using PIWG on the SUN Workstation and the DEC VAX Computer		5. FUNDING NUMBERS PE: 63756 PR: 2853 TA: 01 WU: 03		
6. AUTHOR(S) DAVIDSON, STEVEN A., Major, USAF Reserves				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Avionics Directorate Wright Laboratory Air Force Materiel Command Wright-Patterson AFB OH 45433-7409		8. PERFORMING ORGANIZATION REPORT NUMBER WL-TR-94-1049		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Wright Laboratory, Avionics Directorate Air Force Materiel Command Wright-Patterson AFB OH 45433-7409		10. SPONSORING / MONITORING AGENCY REPORT NUMBER WL-TR-94-1049		
11. SUPPLEMENTARY NOTES Approved for Public Release; Distribution Unlimited				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This report is a laboratory user's guide for evaluating performance of ADA compilers using the Association of Computing Machinery's (ACM) Performance Issues Working Group (PIWG) benchmarking test suite. The report covers the August 1, 1990 version PIWG suite. Part 1 of the report presents a detailed step-by-step instruction for installing these tests on a SUN workstation using a UNIX operating system, and performing an evaluation of a SPARCAda compiler. Part 2 of the report covers installation and execution of costs on the Wright Laboratory WL/AAAF VAX/4000 cluster. Results include recommendations for future work: optimization for avionics' functions by identification, selection, and execution of test subsets appropriate to specific functions, such as navigation, weapon delivery, and electronic warfare; modification of UNIX script files so that execution is completely automatic; computation of averages and variabilities of test results; and adding graphics to show comparisons of various Ada compilers.				
14. SUBJECT TERMS Ada compilers; software performance; Ada benchmark tests; PIWG tests			15. NUMBER OF PAGES 125	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

TABLE OF CONTENTS

Section 1 - INTRODUCTION

1. Background.....	1
2. Purpose.....	1
3. Assumptions.....	1
4. Format Conventions.....	2

Section 2 - RUNNING PIWG ON A SUN WORKSTATION

1. Step 1 - Set up the PIWG Directory.....	3
2. Step 2 - Print the "read.me" File.....	3
3. Step 3 - Copy "compile.ver" to "verdix.sh".....	3
4. Step 4 - Modify "verdix.sh".....	3
5. Step 5 - Install PIWG.....	4
6. Step 6 - Select the Correct CPU_TIME_CLOCK Routine.....	5
7. Step 7 - Add Other Routines to the "sup" Subdirectory.....	5
8. Step 8 - Verify Contents of Subdirectory "sup".....	6
9. Step 9 - Executing the FIRST RUN Tests.....	6
10. Step 10 - Executing the SECOND RUN Tests.....	8
11. Step 11 - Executing the THIRD RUN Tests.....	8

Section 3 - RUNNING PIWG ON THE AAAF VAX

1. Step 1 - Read the PIWG Files to the VAX.....	10
2. Step 2 - Create the PIWG Directory.....	10
3. Step 3 - Copy PIWG Files Into the Work Directory.....	10
4. Step 4 - Go to the PIWG Subdirectory.....	10
5. Step 5 - Modify File "compile.com".....	10
6. Step 6 - Rename file "vaxconfi.com".....	11
7. Step 7 - Execute the PIWG Files.....	11
8. Step 8 - Modify File "zcompile.com" for SECOND RUN Tests.....	11
9. Step 9 - Execute the SECOND RUN Tests.....	11
10. Step 10 - Modify File "z2comp.com" for THIRD RUN Tests.....	12
11. Step 11 - Execute the THIRD RUN Tests.....	12

Section 4 - SUMMARY

1. Observations.....	13
2. Recommendations.....	13

References.....	15
-----------------	----

TABLE OF CONTENTS (Cont.)

APPENDICES

Appendix A - "READ.ME" File.....	16
Appendix B - "VERDIX:SH"	24
Appendix C - PIWG (First) Execution Tests Result Using SPARCAda on the SUN Workstation.....	40
Appendix D - Script File for "SECOND RUN" PIWG Tests on the SUN Workstation.....	55
Appendix E - PIWG "SECOND RUN" Results Using SPARCAda on the SUN Workstation.....	57
Appendix F - Script File for "THIRD RUN" PIWG Tests on the SUN Workstation.....	59
Appendix G - PIWG "THIRD RUN" Results Using SPARCAda on the SUN Workstation.....	63
Appendix H - "COMPILE.COM" File.....	65
Appendix I - PIWG (FIRST) Execution Test Results Using DEC Ada on the VAX Computer.....	69
Appendix J - Command File for "SECOND RUN" PIWG Tests on the VAX Computer.....	103
Appendix K - PIWG "SECOND RUN" Results Using DEC Ada on the VAX Computer.....	106
Appendix L - Command File for "THIRD RUN" PIWG Tests on the VAX Computer.....	111
Appendix M - PIWG "THIRD RUN" Results Using DEC Ada on the VAX Computer.....	115

FOREWORD

This technical report was prepared by Major Steven A. Davidson, USAF Reserves, as part of his 2-week active-duty tour at Wright Laboratory, Avionics Directorate, System Avionics Division, Avionics Logistics Branch (WL/AAAF), Wright-Patterson AFB, OH during the period 7-18 June 1993.

This report documents using the August 1, 1990 version of the PIWG benchmark test suite on a SUN Workstation and the WL/AAAF VAX/4000 cluster. The intent of this report is to provide an easier access to PIWG benchmark test suite for researchers within the Government by providing a "hands-on" experience guide.

The work was accomplished under the Aeronautical Systems Center Air Force Reserves Project 93-451-LAB.

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Section 1 INTRODUCTION

1. Background.

In November 1984, the Association of Computing Machinery (ACM) Special Interest Group on Ada (SIGAda) Performance Issues Working Group (PIWG) was formed. PIWG's charter is to provide performance related information on Ada compilers to the Ada community.

Accordingly, a PIWG team of volunteers created a suite of Ada benchmark test programs in 1985 to serve as a performance measurement tool. Since that time, a new version of the PIWG suite has been produced every year by improving measurement techniques and by adding new test programs. The PIWG suite is freely available and is widely distributed. It has, therefore, become a performance measurement standard in the Ada community. [1]

2. Purpose.

The purpose of this paper is to provide an explicit, step-by-step worked-example of how to install and execute the August 1, 1990 version of the PIWG suite. Section 2 describes how to do this on a SUN Workstation running under the UNIX operating system using the new SPARCAda compiler. Section 3 covers installation and execution of the PIWG suite on a Digital Equipment Corporation (DEC) VAX main frame computer.

3. Assumptions.

This paper makes the following assumptions:

- a. **PIWG Suite.** The August 1, 1990 versions of the PIWG suite of benchmark test programs, associated script files, .COM files, and "read.me" file are used in this report.
- b. **UNIX Commands.** The reader is not required to know the UNIX command language used on the SUN Workstation. Information to perform essential tasks to execute PIWG on the SUN Workstation is provided in Section 2.
- c. **DEC Command Language (DCL).** The reader is not required to know the DCL used on the VAX computer. Information to perform essential tasks to execute PIWG on the VAX computer is provided in Section 3.
- d. **Ada Compilation System (acs).** The .COM files from the PIWG distribution tape or bulletin board are written such that the Ada compiler, linker, and library are invoked by means of the VAX DCL utility "acs". This utility software must be available on the VAX machine being used to compile the PIWG suite.

e. **Text File Editors.** The reader is expected to be able to use a text editor to modify script files on the SUN Workstation and command files (.COM) on the VAX computer. For the VAX computer, screen editors such as EDT or the Language Sensitive Editor (LSE) are recommended.

4. Format Conventions.

This paper will use the following format conventions to promote clarity and minimize confusion:

a. **Interactive Commands.** Commands that are to be entered on the SUN Workstation or on a VAX terminal are indented and set apart from the supporting text by blank lines. Each command appears on a separate line. The end of the line implies the carriage return (RETURN) key is to be pressed.

b. **File Names.** File names and directory names discussed in the test will be enclosed within double-quote marks (example: "read.me").

c. **Comments.** In Section 2, comments associated with inputs required for script files will be preceded by two hyphens (--), per the Ada convention. Do not enter the hyphens or any text that follows on the same line.

Section 2

RUNNING PIWG ON A SUN WORKSTATION•

1. Step 1 - Set up the PIWG Directory

A separate subdirectory should be created on the SUN Workstation to contain the PIWG files. The subdirectory should be created in a partition that has sufficient memory for the PIWG source files and files created as a result of compiling and executing the PIWG files. For the SUN Workstation that produced the PIWG results discussed in this report, the PIWG source files were read into the subdirectory:

```
/home/corbeaux/davidson/8_1_90piwg
```

The PIWG source files were initially in a single compressed file, "8_1_90piwg.tar.z". This file was uncompressed using the UNIX command:

```
uncompress 8_1_90piwg.tar.z
```

This produced the uncompressed file, "8_1_90piwg.tar". The separate PIWG files were extracted using the UNIX command:

```
tar -xvf 8_1_90piwg.tar
```

As a result, numerous PIWG files were produced in the subdirectory "/home/corbeaux/davidson/8_1_90piwg", with the majority having the file extension of ".ada".

2. Step 2 - Print the "read.me" File.

The file "read.me" should be one of the files now available after completing Step 1. Print this file and read it carefully. It gives guidance on executing the PIWG benchmark tests. A copy of the "read.me" file is in Appendix A. The steps that follow refer to the text of this file.

3. Step 3 - Copy "compile.ver" to "verdix.sh"

Copy file "compile.ver" into a new file, "verdix.sh", using the UNIX command:

```
cp compile.ver verdix.sh
```

4. Step 4 - Modify "verdix.sh"

Files "compile.ver" and "verdix.sh" contain the Bourne shell script for building and executing PIWG files under UNIX. However, they also contains explanatory text at the

beginning and near the end of the file that must be removed before the file can be executed as a shell script. Use a text file editor to remove all lines of file "verdix.sh" prior to the first line of the shell script:

```
#!/bin/sh
```

Remove all lines after the last lines of the shell script:

```
done
cd $cur_wd
done
```

Save the modified file as file "verdix.sh". A copy of file "verdix.sh" is in Appendix B.

5. Step 5 - Install PIWG

Install PIWG on the SUN Workstation using the "verdix.sh" script file. First, check you are located in the proper directory by using the UNIX command:

```
pwd
```

The response, "/home/corbeaux/davidson/8_1_90piwg" should appear on the screen. Also check the "verdix.sh" file is in this subdirectory by using the UNIX command:

```
ls verdix.sh
```

The response should echo back the file name, "verdix.sh". Make appropriate corrections as necessary (or get help) before proceeding. Start execution of the script file by entering the UNIX command:

```
sh verdix.sh
```

After execution begins, the shell will ask for a series of inputs. Respond as follows for initial installation:

```
1 -- to select Install PIWG
/home/corbeaux/davidson/8_1_90piwg
-- to enter path for desired Verdix PIWG location
1 -- for System V timing procedure
1 -- select Self-Target
```

The "verdix.sh" shell will create new subdirectories for each of the various test groups that are described under the heading, "FIRST OF EXECUTION TESTS" in file "read.me" (see Appendix A). These subdirectories are "a", "b", "c", "d", "e", "f", "g", "h", "i", "p", "t", and "y". The shell will also create the subdirectory "sup" for the basic routines in the program library for

execution timing tests. The shell will additionally create subdirectories "z" and "zother" for the "SECOND RUN" and "THIRD RUN" tests, respectively, described in the "read.me" file. These are discussed later in Paragraphs 10 and 11 in Section 2 of this report.

The shell will rename all Ada files with an ".a" extension (instead of the original ".ada" extension) and copy them into their appropriate subdirectory.

6. Step 6 - Select the Correct CPU_TIME_CLOCK Routine

During the installation, "verdix.sh" will write "a000019.a" to subdirectory "sup" if System•V is selected or "a000018.a" if VAX/UNIX or SUN3 UNIX is selected. No other options are available. Neither "a000018.a" nor "a000019.a" allow test "y000002.a" to execute correctly when run interactively. Test "y000002.a" is a check for CPU_TIME_CLOCK. As the "read.me" file indicates, "a000017a.a" is the correct routine to use with the Verdix Ada. Until the "compile.ver" file (and the resulting "verdix.sh" shell) is corrected, the user must copy the correct routine for CPU_TIME_CLOCK into subdirectory "sup" by doing the following:

Starting in subdirectory "/home/corbeaux/davidson/8_1_90piwg", copy "a000017a.ada" into the "sup" subdirectory and rename it as "a000017a.a" by using the UNIX command:

```
cp a000017a.ada ./sup/a000017a.a
```

Go to the "sup" subdirectory:

```
cd sup
```

Remove file "a000019.a" that was written to subdirectory "sup" by the "verdix.sh" shell:

```
rm a000019.a
```

Verify that "a000017a.a" replaced file "a000019.a" by listing the files in subdirectory "sup":

```
ls
```

Return to the next higher subdirectory, "/home/corbeaux/davidson/8_1_90piwg" for proper execution of the remaining steps:

7. Step 7 - Add Other Routines to the "sup" Subdirectory

There are three other basic routines that MUST be added to the "sup" subdirectory for proper test execution. The "compile.ver" file should be corrected so this will be done automatically during installation. Until the "compile.ver" file (and the resulting "verdix.sh" shell) is corrected, the user must copy and rename these files manually by entering the following UNIX commands (while in subdirectory /home/corbeaux/davidson/8_1_90piwg):

```
cp a000047.ada ./sup/a000047.a
```

```
cp a000048.ada ./sup/a000048.a
cp a000051.ada ./sup/a000051.a
```

8. Step 8 - Verify Contents of Subdirectory "sup"

Subdirectory "/home/corbeaux/davidson/8_1_90piwg/sup" should contain the following files as a result of completing Steps 5 through 7:

```
GVAS_table
a000001.a
a000017a.a
a000021.a
a000022.a
a000031.a
a000032.a
a000041.a
a000042.a
a000047.a
a000048.a
a000051.a
ada.lib
gnrx.lib
```

Although some additional files may be present (such as "ada.lib%", etc.), it is important that the files "a000011.a" through "a000019.a" are NOT in subdirectory "sup", except for file "a000017a.a". Refer to Paragraph 1 of the "read.me" file (Appendix A), and the comments that accompany the listing of files in Paragraph 3 of the "read.me" file.

9. Step 9 - Executing the FIRST RUN Tests

Execute the PIWG files for the FIRST RUN as described in the "read.me" file by using the "verdix.sh" script file. First, check you are located in the proper directory by using the UNIX command:

```
pwd
```

The response, "/home/corbeaux/davidson/8_1_90piwg" should appear on the screen. Also check the "verdix.sh" file is in this subdirectory by using the UNIX command:

```
ls verdix.sh
```

The response should echo back the file name, "verdix.sh". Make appropriate corrections as necessary (or get help) before proceeding. Start execution of the script file by entering the UNIX command:

```
sh verdix.sh
```

After execution begins, the shell will ask for a series of inputs. Respond as follows:

```
3      -- to run PIWG
/home/corbeaux/davidson/8_1_90piwg
      -- to enter path for desired Verdix PIWG location
y      -- to run test WITHOUT suppressed checking
n      -- to say "no", support library is not compiled
<CR>   -- choose the default for output to screen
2      -- choose tests by letter (test group letter)
a      -- choose the "a" tests (only)
<CR>   -- (blank line signals no more tests follow)
```

The "verdix.sh" shell will immediately begin compiling, linking, and executing the tests selected. In this case, the test routines in Group "a", as described in the "read.me" file will be executed. For the first execution and after ANY changes to the files in subdirectory "/home/corbeaux/davidson/8_1_90piwg/sup", ALWAYS respond with "n" for "no" when asked if the support library is compiled. If you have any doubts, choose "n" in response to this question. If the library was previously compiled and "n" is selected, a lot of warning messages ("id hides outer definition") will be generated. This will not adversely affect execution of PIWG. However, if ANY changes are made to the files in subdirectory "/home/corbeaux/davidson/8_1_90piwg/sup" and the support library is NOT compiled by selecting "y" for "yes", results are unpredictable and errors are likely to occur.

Output to the screen is selected because this allows one to verify that the PIWG tests are executing. However, it is also possible to get a hard copy of these results on paper for off-line review and analysis. With the SUN Workstation, the screen output is stored in an output buffer file. Using the mouse, one can scroll back to the beginning of the screen output and, using block marking procedures, copy the PIWG output portion from the screen buffer file to a separate user file for subsequent printing. For details on how to do this, get help from someone experienced with using the SUN Workstation.

When the "a" test group is executed, the last test in the group, Routine "a000095.a", expects to read the output from a script file as an input file to produce a compressed file of results. I could not get this to work but entering a carriage return causes an error, the program is abandoned, and execution stops. (Routine "a000095" is not addressed in the "read.me" file; its purpose is unknown.)

To run all tests of the PIWG, initiate execution of the script file again by entering the UNIX command:

```
sh verdix.sh
```

After execution begins, the shell will ask for a series of inputs. Respond in a similar manner as described for the previous example except choose "all tests" instead of "tests by letter". A copy of the results of choosing all tests is in Appendix C.

10. Step 10 - Executing the SECOND RUN Tests

Execute the PIWG files for the SECOND RUN as described in the "read.me" file by using the "second" script file that was created during the installation of PIWG (Step 5 of this Section). Appendix D contains a copy of this script file. Move to the "/home/corbeaux/davidson/8_1_90piwg/z" subdirectory from the "/home/corbeaux/davidson/8_1_90piwg" subdirectory by using the UNIX command:

```
cd z
```

Initiate execution of the script file by entering the UNIX command:

```
sh second
```

Results from the SECOND RUN are provided in Appendix V. As the results state, the difference between the start and stop wall times is the bench mark value of interest. Appendix V shows this bench mark value for the SUN Workstation running "SPARCAda" is (62140 - 61811 =) 329 seconds.

11. Step 11 - Executing the THIRD RUN Tests

Execute the PIWG files for the THIRD RUN as described in the "read.me" file by using the "third" script file that was created during installation of PIWG (Step 5 of this Section). Appendix VI contains a copy of this script file. Move up to the "/home/corbeaux/davidson/8_1_90piwg" subdirectory from the "/home/corbeaux/davidson/8_1_90piwg/z" subdirectory by using the UNIX command:

```
cd..
```

Move down to the "/home/corbeaux/davidson/8_1_90piwg/zother" subdirectory by using the UNIX command:

```
cd zother
```

Initiate execution of the script file by entering the UNIX command:

```
sh third
```

Results from the THIRD RUN are provided in Appendix VII. As the results state, the difference between the start and stop wall times are the bench mark values of interest. Appendix VII shows these bench mark values for the SUN Workstation running "SPARCAda" are:

compiling files	Stop Wall Time 37681 seconds
separately	<u>Start Wall Time 37159 seconds</u>
	Total Time 522 seconds

compiling files at	Stop Wall Time 38496 seconds
once in z000200.a	<u>Start Wall Time 37681 seconds</u>
	Total Time 815 seconds

Section 3

RUNNING PIWG ON THE AAAF VAX

1. Step 1 - Read the PIWG Files to the VAX

Read the August 1, 1990 version of the PIWG files into a public directory that has read-only protection. Ask your VAX System Manager to create a public directory and read the PIWG files into it. The PIWG files were read into public directory \$1\$DIA1:[PIWG] on the AAAF VAX.

2. Step 2 - Create the PIWG Directory

A separate subdirectory should be created in the user's area to contain the PIWG files. The remaining steps in this paper for executing the PIWG files are based on the following directory and subdirectory structure. The top directory is [davidson] on Disk 9 (\$1\$DIA9:). Of course, the reader's VAX account name would be substituted for "davidson" in all the commands presented in this paper. The work subdirectory is set up by entering the VAX command:

```
create/dir [davidson.piwg]
```

Subdirectory [davidson.piwg] is the PIWG working directory. This is where all PIWG files are modified and executed.

3. Step 3 - Copy PIWG Files Into the Work Directory

Copy the PIWG files into your directory by entering the VAX command:

```
copy $1$dia1:[piwg]*.* $1$dia9:[davidson.piwg]*.*
```

The above command is based on the public directory being located on Disk 1 (\$1\$DIA1).

4. Step 4 - Go to the PIWG Subdirectory

Move to the PIWG subdirectory before performing the remaining steps. Enter the VAX Command:

```
set def [davidson.piwg]
```

5. Step 5 - Modify File "compile.com"

Using an ASCII text file editor, change the sixth line of this file from:

```
$ SET DEF $1$DIA1:[PIWG]
```

to:

```
$ SET DEF $1$DIA9:[DAVIDSON.PIWG]
```


6. Step 6 - Rename file "vaxconfi.com"

Change the name of this file to agree with the file name specified in the tenth line of the "compile.com" file:

```
$ @ VAXCONFIG
```

Enter the VAX Command:

```
rename vaxconfi.com vaxconfig.com
```

(Alternately, the tenth line of the "compile.com" file could be modified to agree with the file name "vaxconfig.com".)

7. Step 7 - Execute the PIWG Files

Submit file "compile.com" as a batch job to prevent the terminal from being unusable during the rather long execution time (about 20 minutes) and to have the output written to a file, "piwg.log". Appendix VIII is a copy of the "compile.com" file after the appropriate modification were made. Enter the following VAX command:

```
submit/notify/log_file=[davidson.piwg]piwg.log compile.com
```

Print the resulting file "piwg.log" and review it. A copy of the PIWG results on the VAX is in Appendix A.

8. Step 8 - Modify File "zcompile.com" for SECOND RUN Tests

File "zcompile.com" is provided on the distribution tape and can be used to execute the SECOND RUN tests (described in the "read.me" file) on the VAX computer. Prior to using this command file, the "\$ SET DEF" line must be modified, as was done in Step 5 for the "compile.com" file. Change the third line from:

```
$ SET DEF $1$DIA1:[PIWG]
```

to:

```
$ SET DEF $1$DIA9:[DAVIDSON.PIWG]
```

9. Step 9 - Execute the SECOND RUN Tests

Submit file "zcompile.com" as a batch job to have the output written to a file, "run2.log". Appendix X is a copy of the "zcompile.com" file after the modification in Step 8 was completed. Enter the following VAX command:

```
submit/notify/log_file=[davidson.piwg]run2.log zcompile.com
```

Print the resulting file "run2.log" and review it. A copy of the SECOND RUN results on the VAX is in Appendix XI.

10. Step 10 - Modify File "z2comp.com" for THIRD RUN Tests

File "z2comp.com" is provided on the distribution tape and can be used to execute the THIRD RUN tests (described in the "read.me" file) on the VAX computer. Prior to using this command file, the "\$ SET DEF" line must be modified, as was done in Step 5 for the "compile.com" file. Change the third line from:

```
$ SET DEF $1$DIA1:[PIWG]
```

to:

```
$ SET DEF $1$DIA9:[DAVIDSON.PIWG]
```

11. Step 11- Execute the THIRD RUN Tests

Submit file "z2comp.com" as a batch job to have the output written to a file, "run3.log". Appendix XII is a copy of the "z2comp.com" file after the modification in Step 8 was completed. Enter the following VAX command:

```
submit/notify/log_file=[davidson.piwg]run3.log z2comp.com
```

Print the resulting file "run3.log" and review it. A copy of the THIRD RUN results on the VAX is in Appendix XIII.

Section 4 SUMMARY

1. Observations

The PIWG command files for the DEC VAX computer are very easy to use. They only require a simple change to specify the appropriate directory in the user's account. The one error I found concerning "vaxconfig.com" file (see Section 3, Step 6) is non-fatal but should be corrected so that test results will always include useful information about the VAX machine and version of DEC Ada being used.

The PIWG UNIX script files for the SUN Workstation (or other UNIX-based computers) need improvement. The user must manually perform many file manipulations prior to running the script file to obtain useful results.

A brief comparison of the PIWG (FIRST) Execution Tests between the VAX and the SUN Workstation shows that many more test results are available for the VAX version of PIWG (printed on 41 pages in Appendix A) than for the SUN version (printed on 16 pages in Appendix C). A comparison of CPU times for the PIWG tests performed by both machines shows the SUN Workstation is much faster than the VAX...more than 9 times faster for some tests!

The SECOND RUN test is a compile time benchmark. As the script file comments state, it is meaningful only when run with no other tasks on the machine. Consequently, the result of 329 seconds for the SUN Workstation (see Section 2, Step 10) is meaningful but the value of 966 seconds (42445 - 41479) for the VAX (see Appendix XI) is not reliable because the VAX was always operating as a time-share machine with multiple users when the PIWG tests were done. The SECOND RUN results for the VAX are provided only to show the format of the results.

This same observation applies to the THIRD RUN tests. However, Section 2, Step 11, shows that it is faster for the SUN Workstation to compile programs in separate files (522 seconds) than to compile the same number of programs lumped together in a single file (815 seconds).

2. Recommendations

This paper can serve as a starting point for additional projects. Some examples are as follows:

a. Determine which PIWG tests are most important for Air Force avionic applications. PIWG avionic tests should be further subdivided into navigation, flight control, weapon delivery, or electronic countermeasure tests, for example. Write modified UNIX script files or VAX command files that execute a subset of PIWG tests for a specific application. These modified script files could help the user focus a PIWG evaluation on problems most important to the Air Force.

b. Improve the UNIX script files so that execution is completely automatic; no manual file manipulations are necessary.

c. Perform multiple PIWG runs on a single machine and compute averages and variations of tests results.

d. Perform PIWG runs on different machines and do a detailed comparison of the test results between the machines. Show these differences graphically on charts.

References

1. "A Rationale for the Design and Implementation of Ada Benchmark Programs," Ada Letters Special Edition (Association for Computing Machinery, Inc., 11 West 42nd Street, New York, N.Y. 10036), Volume X, Number 3, Winter 1990.

APPENDIX A

"READ.ME" FILE

APPENDIX A README FILE

README

There are three complete runs to be made on each computer/compiler combination. The first run makes execution performance measurements on Ada features and composite benchmarks. The second run makes a composite compile speed measurement including linking and execution. The third run compiles about 100 files from the SIGAda programming contest winner, then compiles the same source code again as one file.

For your convenience, scripts are provided for several computer/compiler combinations. The scripts are needed because there are multiple bodies for some of the Ada package specifications.

**DO NOT USE A " MAKE " ON THE ENTIRE DIRECTORY !!!
DO NOT USE " SUPPRESS " except where indicated !!!**

A sample output from all three runs is shown in SAMPLE.OUT. Sample log files are shown as *.LOG

FIRST RUN

The first run is typified by COMPILE.COM (VAX) , COMPILE.CLI (ROLM/Data General) or COMPILE.BAT (PC compatibles).

COMPILE.VER (Verdix on UNIX). If necessary, build a script from one of these for your computer/compiler. The script is intended to compile a selection of the Ada source, link them into one or more executable programs, and then execute the tests and print a report. Please send PIWG the minimum repeatable execution times. Execute tests individually if times differ significantly from executing the combined tests. In order for tests to fit on some embedded computers, every test must be linked and downloaded separately. AEMBEDDED.COM along with LINKD.COM is a sample script for preparing absolute images for down loading. (These must be tailored for each host/compiler/target). For embedded computers, executions must be on real hardware. These measurements are not suitable for simulators. COPY.RAT may help selecting the files for the Rational R1000.

If running on a PC, get rid of the .COM files. They will cause the PC to hang if you try to use a .BAT file of the same name. Various .BAT files are provided to compile, link and execute each test individually.

Generally, compilation order is alphabetical, but DO NOT COMPILE EVERYTHING. There are a number of choices to be made depending on the computer/compiler being used.

The test suite is designed for running with the predefined type INTEGER being 32 bits. The Uniformity Rapporteur Group, URG, Uniformity Issue, UI-0008, states in part that an implementation should have a mode of operation in which INTEGER'LAST $\geq 2^{31}-1$. Most tests will work with 16 bits for type INTEGER. Only a few tests will fail, turn in results even if some tests fail. A few tests are quite large. Turn in results even if these large tests do not compile or execute.

Most tests do not include TEXT_IO. In an embedded computer that, for some reason, can not handle TEXT_IO, do not run the "G" tests. Remove "with TEXT_IO" and calls on PUT. Try something to be sure the code to be measured is not optimized away. This applies to tests:

APPENDIX A READ.ME FILE

A000090, A000091, A000092, A000093, B000010, Y000001.

1. Choose one of A000011 through A000019 for CPU_TIME_CLOCK. This is computer/operating system dependent. A000011 gets only WALL time via the package CALENDAR. We hope all compilers can do better than this. A tailored version that gives better accuracy is allowed. Please send an electronic copy in with your results.
2. Choose one of A000042 through A000044 for PIWG_IO body. A000042 produces printout as it runs (only one sample per run). Please direct the printout to disk and send in an electronic copy, that have no TEXT_IO. (Some editing will be required.)
3. Choose from making each procedure A000090 through Y000003 a main program, or using A000100 as a single main program with A000042, or using A000101 and A000102 as a pair of smaller main programs, or using a script like AEMBEDDED.COM, or COMPILE.BAT with each test as an individual main program.

The first group of files below establishes the basic routines in the program library for the execution timing tests. The complete test suite can be compiled, linked and run from one library. All files are of the form NAME.TYP with TYP being ADA for all Ada source files.

ADA A000001	DURATION_IO instantiation	
ADA A000011	CPU_TIME_CLOCK.ADA	PICK ONE from this set or do your own
ADA A000012	CPU_TIME_CLOCK.VAX	-----
ADA A000013	CPU_TIME_CLOCK.DG	(manufacturers should help)
ADA A000014	CPU_TIME_CLOCK.UNIX	worked on Gould HZ = 60
ADA A000014A	CPU_TIME_CLOCK.UNIX	UNIX with HZ = 100
ADA A000015	CPU_TIME_CLOCK.R1000	
ADA A000016	CPU_TIME_CLOCK.ULTRIX	
ADA A000017	CPU_TIME_CLOCK.UNIX	worked on SUN
ADA A000017A	CPU_TIME_CLOCK.UNIX	Verdix
ADA A000018	CPU_TIME_CLOCK.Meridian	
ADA A000019	CPU_TIME_CLOCK.RR Janus	
ADA A000021	BREAK	optimization control package spec
ADA A000022	BREAK	optimization control package body
ADA A000031	ITERATION	Iteration control package spec
ADA A000032	ITERATION	Iteration control package body
ADA A000041	PIWG_IO	output package spec (universal)
		PICK ONE from group below:
ADA A000042	PIWG_IO	package body for screen/printer output
ADA A000044	PIWG_IO	package body for save in memory, no TEXT_IO
ADA A000047	PIWG_TIMER_GENERIC	spec
ADA A000048	PIWG_TIMER_GENERIC	body

THE A000051 ..A000055 ARE FOR THE HOST ONLY! Do not compile these for an embedded execution target.

ADA A000051	A000051	executable procedure to print WALL and
-------------	---------	--

APPENDIX A READ.ME FILE

ADA A000052	A000052	CPU time (only for HOST, not embedded target) A set of 4 executable procedures that can be used to measure CPU and Wall time without instrumenting the run begin measured. Place optional control between A000052 and A000053. Place test being measured between A000054 and A000055.
ADA A000053	A000053	
ADA A000054	A000054	
ADA A000055	A000055	
ADA A000095	A000095	A program to read a log file from COMPILE.* and produce a compact listing PIWGTBL.
ADA A000098		This is a skeleton procedure that can be copied and edited to construct more tests that have multiple parts or multiple copies of the same test. DO NOT COMPILE THIS. It is for editing to make more tests.
ADA A000099		This is a skeleton procedure that can be copied and edited to construct more tests. DO NOT COMPILE THIS. It is for editing to make more tests.
ADA A000100	A000100	This is a top level procedure that calls all the other executable timing tests. It is useful if there is a "MAKE" facility available. (This may be too big to execute on many computers. Tests may be run individually or this may be split into smaller sets.)
ADA A000101		This and A000102 are A000100 split in two pieces. It includes tests A000090 .. 000004.
ADA A000102		This and A000101 are A000100 split in two pieces. It includes tests E000001 .. 000001. Split up groups of tests as necessary to get them run.

FIRST OF EXECUTION TESTS

ADA A000090	Measure clock resolution by second differences
ADA A000091	DHRYSTONE
ADA A000092	WHETSTONE using manufacturers math routines (must edit)
ADA A000093	WHETSTONE using standard Ada math routines
ADA A000094A..K	HENNESY benchmarks

This group of tests measures a tracker algorithm.

**APPENDIX A
READ.ME FILE**

**ADA B000001A
ADA B000001B**

Use pragma SUPPRESS or NOCHECK or
other switch

**ADA B000002A
ADA B000002B**

Use pragma SUPPRESS or NOCHECK or
other switch

**ADA B000003A
ADA B000003B**

Use pragma SUPPRESS or NOCHECK or
other switch

**ADA B000004A
ADA B000004B**

Use pragma SUPPRESS or NOCHECK or
other switch

**ADA B000010
ADA B000011
ADA B000013**

This group of tests measures task creation related timing.

**ADA C000001
ADA C000002
ADA C000003**

This group of tests measures dynamic elaboration related timing.

**ADA D000001
ADA D000002**

These may not run on some targets. DO
NOT CHANGE THEM

**ADA D000003
ADA D000004**

This group of tests measures exception related timing.

**ADA E000001
ADA E000002
ADA E000003
ADA E000004
ADA E000005**

This group of tests measures coding style related timing.

**ADA F000001
ADA F000002**

This group of tests measures TEXT_IO related timing.

**ADA G000001
ADA G000002**

SOME MAY NOT RUN ON SOME
TARGETS.

**ADA G000003
ADA G000004
ADA G000005
ADA G000006
ADA G000007**

**APPENDIX A
READ.ME FILE**

This group of tests measures chapter 13 related features.

ADA H000001
ADA H000002
ADA H000003
ADA H000004
ADA H000005
ADA H000006
ADA H000007
ADA H000008
ADA H000009

This group of tests measures loop overhead related timing.

ADA L000001
ADA L000002
ADA L000003
ADA L000004
ADA L000005

This group of tests measures procedure call related timing.

ADA P000001
ADA P000002
ADA P000003
ADA P000004
ADA P000005
ADA P000006
ADA P000007
ADA P000010
ADA P000011
ADA P000012
ADA P000013

This group of tests measures task related timing.

ADA T000001
ADA T000002
ADA T000003
ADA T000004
ADA T000005
ADA T000006
ADA T000007

ADA T000008

This is a check on T000001 that only works
on some machines

This group of tests measures delay related timing.

ADA Y000001
ADA Y000002

This takes several minutes to run
Run interactively to check
CPU_TIME_CLOCK

**APPENDIX A
READ.ME FILE**

ADA Y000003

run interactively to check
CALENDAR.CLOCK

The file **COMPILE.COM** is a sample script that compiles and runs the above tests when the host computer is also the target. Commercial computers, software engineering workstations

The file **COMPILE.CLI** is a sample script for Data General MV series for above.

The file **COMPILE.VER** is a sample script for Verdex compiler on Unix.

The file **COMPILE.BAT** is a sample script for a PC. It has associated files **COMP.BAT** and **RUN.BAT** that may need modification. **COMPLMER.BAT** is tailored for Meridian on a PC

The file **AEMBEDDED.COM** is a sample script that compiles the above tests for later execution on an embedded computer.

SECOND RUN

This is a composite compile time measurement.

The second run is typified by **ZCOMPILE.COM** (VAX) or **ZCOMPILE.CLI** (ROLM/Data General) or **ZCOMPILE.BAT** (PC compatibles). Build the necessary script from one of these for your computer/compiler. This script provides one compilation time measurement for the time to compile, link and execute two programs. The execution time is very small compared to compile time. The "execution" for this run can be performed on a simulator (the simulation time is not counted as part of the total time)

The "Z" tests are for measuring compilation time using **A000051**. **A000052 .. A000055** do a calibration and differencing. The execution part of this test may be omitted, but use it for checking to see that executable code was produced and that it will execute properly.

The files, **Z000001** through **Z000023** are all part of one test, see sample script **ZCOMPILE.COM**. See **ZCOMPILE.CLI**, **ZCOMPILE.BAT** for other sample scripts. **ZCOMPILE.LOG** is a sample output.

Compile these files as given. Do not combine in one big file. We have data on these compilations from 1984. We want to be able to plot a five year industry trend. **DON'T CHANGE THE TESTS.**

```
RUN  A000051
RUN  A000051    ! calibrate time to measure time
RUN  A000051
ADA  Z000001    ! FLTIO
ADA  Z000002    ! REFUNCT
ADA  Z000003    ! PREAL
ADA  Z000004    ! PUBASIC
ADA  Z000005    ! PUMECH
ADA  Z000006    ! PUELEC
ADA  Z000007    ! PUOTHER
ADA  Z000008    ! MKSPMECH
ADA  Z000009    ! MKSPELEC
ADA  Z000010    ! PCONSTANT
```

APPENDIX A READ.ME FILE

```

ADA Z000011 ! PUOBASIC
ADA Z000012 ! PUOMECH
ADA Z000013 ! PUOELEC
ADA Z000014 ! PCCONST
ADA Z000015 ! PUCONV
ADA Z000016 ! PUCMKS spec
ADA Z000016A ! PUCMKS body (rather large)
ADA Z000017 ! PUCENGL spec
ADA Z000017A ! PUCENGL body (rather large)
ADA Z000018 ! PHYSICS1
LINK Z000018 ! the linking time is part of the test time
RUN Z000018 ! very short, just check to be sure it really runs
               ! the timing of this is not counted for an embedded computers
               ! it should be run just to be sure it was really compiled

ADA Z000020 ! GENPREAL
ADA Z000021 ! ALLSTMT
ADA Z000022 ! GENSORTSH
ADA Z000023 ! GENSHELLI
LINK Z000023 ! the linking time is part of the test time
RUN Z000023 ! very short, just check to be sure it really runs
               ! the timing of this is not counted for an embedded computers
               ! it should be run just to be sure it was really compiled

RUN A000051 ! final time measurement
               ! when this test can be run with no other tasks running, it
               ! represents a composite software development benchmark

```

THIRD RUN

This is a comparison of compilation speed using each compilation unit in its own file vs. all compilation units in one large file. The files are from the winner of the SIGAda programming contest. This may represent a sample of a small real time program. The programming style may be considered OOP. There is heavy usage of "is separate".

```

RUN A000051 ! initial wall time
ADA Z000100
ADA Z000101
...
ADA Z000199 102 files including 106A and 106B
RUN A000051 ! final wall time, compute difference and report
RUN A000051 ! initial wall time
ADA Z000200 ! this includes all files Z000100 .. Z000199
RUN A000051 ! final wall time, compute difference and report

```

PLEASE use common sense in analyzing results.

APPENDIX B

"VERDIX.SH" FILE

APPENDIX B
"VERDIX.SH" FILE

```
#!/bin/sh
# verdix.run -- Bourne shell script for building and executing PIWGs under UNIX.
n='-n'
c='\c'
contains=`echo "Contains \c" | grep c`
case $contains in
    "") n="";;
    *) c="";;
esac
TRUE=1
echo "This is the Verdix PIWG execution shell file. This file will do the following: Copy the"
echo "tests into separate sub-directories, with the .a suffix. Select and compile the appropriate"
echo "A files for subsequent test execution. Allow you to select the host-target combination"
echo "that you wish to test. Allow you to run selected tests, or the entire suite."
echo ""
while [ $TRUE ]
do
    echo "Which operation do you wish to perform:"
    echo "1) Install PIWG's for Verdix execution"
    echo "(This will need to copy all test files to a new location)"
    echo "2) Modify tests to use Verdix-Specific Pragmas"
    echo "3) Run PIWG's"
    echo ""
    echo $n "Enter choice : $c"
    read option
    case $option in
        [1-3]) break;;
        *) echo "valid choices are 1, 2, and 3"; echo "";;
    esac
done

echo "Enter Path to Desired Verdix PIWG Location"
echo $n "path : $c"
read pbd
foo=`echo $pbd | wc -w`
if [ $foo = 0 ]
then
    pbd='.'
    elif [ ! -d $pbd ]
    then
        mkdir $pbd
fi
cur_wd=`pwd`
if [ $option = 1 ]
then
```

APPENDIX B
"VERDIX.SH" FILE

```
echo ""
echo "You now need to select the host-target combination that you will use. This
echo "selection will determine the timing support routine selected. If you are going to"
echo "run on a cross-target, you will need to perform rather substantial installation and"
echo "configuration procedures to run the suite. Please refer to the VADS manual"
echo "under 'Installation and Maintenance' and 'Getting Started'."
while [ $TRUE ]
do
    echo ""
    echo "Choices for timing procedure:"
    echo "  1 - System V"
    echo "  2 - VAX/UNIX"
    echo "  3 - Sun3 UNIX"
    echo "  4 - Sun3 Cross to 68020"
    echo "  q - Quit and Exit"
    echo ""
    echo $n "Enter Choice : $c"
    read choice
    case $choice in
        [1-4]) break;;
        [qQ]) exit;;
        *) echo ""; echo "Valid choices are 1-4"; echo ""; continue;;
    esac
done
if [ ! -d $pbd/sup ]
then
    if [ $choice = 4 ]
    then
        echo "You need to enter the path to the user configuration directory. If you"
        echo "have not created and configured this directory, then you are not ready to"
        echo "run the suite. Consult the VADS manual under 'Installation and"
        echo "Maintenance'."
        echo ""
        echo $n "User Config Dir : $c"
        read conf_dir
        conf=`echo $conf_dir | wc -w`
        if [ $conf = 0 ]
        then
            echo "Please re-run the script later when this directory is available."
            exit
        else
            cd $conf_dir
            option=`grep 'OPTIONS:INFO' ada.lib | wc -w`
            lblock=`grep 'LINK_BLOCK' ada.lib | wc -w`
            if [ $option -eq 0 -o $lblock -eq 0 ]
            then
```


APPENDIX B "VERDIX.SH" FILE

```

echo "Your user configuration library does not have all of the elements"
echo "necessary to run the suite using this script. It needs to have a"
echo "LINK_BLOCK directive as well as a linker options file. This"
echo "directive as well as a linker options file. This script is set up to"
echo "run the tests on a cross target assuming that these directives are"
echo "available in that library. Refer to the VADS manual for"
echo "information on what these are, and how to create them."
fi
cd $cur_wd
fi
a.mklib $pbd/sup $conf_dir
else
a.mklib -i $pbd/sup
fi
pathlist=`grep ADAPATH $pbd/sup/ada.lib`
standard=`echo $pathlist | awk '{ n=split($0,a," ");
for (i=n;i>0;i--) print a[i]}' | grep "standard"`
vads_loc=`echo $standard | sed -n '1,$x/standardxxgp`
if [$choice = 4]
then
cd $pbd/sup
a.path -a $vads_loc/cross_io
cd $vads_loc
cd $pbd/sup
hw_chk=`echo $vads_loc | grep 68881 | wc -w`
echo $hw_chk
if [$hw_chk = 0]
then
echo "It is probable that test A000092 will not compile,"
echo "because a non-hardware version of ATAN is not"
echo "provided. If you wish to run this test, you will need to"
echo "provide this function."
echo ""
else
cat > math.a << --endmath

package MATH is

-- Math interfaces to MATH for use with A00092.a in PIWG

function SIN (X : FLOAT) return FLOAT;
function COS (X : FLOAT) return FLOAT;
function ATAN (X : FLOAT) return FLOAT;
function SQRT (X : FLOAT) return FLOAT;
function EXP (X : FLOAT) return FLOAT;
function LOG10 (X : FLOAT) return FLOAT;
function LOG (X : FLOAT) return FLOAT;

```

APPENDIX B
"VERDIX.SH" FILE

```
pragma INLINE(SIN, COS, ATAN, SQRT, EXP, LOG10, LOG);  
end MATH;
```

```
with MACHINE_CODE;  
package body MATH is
```

```
-- use coprocessor instructions to perform operations
```

```
    FLOAT_SIZE : constant := 64;  
    SIZE_ASSERTION_ERROR : exception;
```

```
procedure SIN_68881(X : in FLOAT; RESULT : out FLOAT) is
```

```
    use MACHINE_CODE;
```

```
begin
```

```
    code_2' (FMOVE_D, X'REF, FP0);  
    code_1' (FSIN_D, FP0);  
    code_2' (FMOVE_D, FP0, RESULT'REF);  
end SIN_68881;
```

```
procedure COS_68881(X : in FLOAT; RESULT : out FLOAT) is
```

```
    use MACHINE_CODE;
```

```
begin
```

```
    code_2' (FMOVE_D, X'REF, FP0);  
    code_1' (FCOS_D, FP0);  
    code_2' (FMOVE_D, FP0, RESULT'REF);  
end COS_68881;
```

```
procedure ATAN_68881 (X : in FLOAT; RESULT : out FLOAT) is
```

```
    use MACHINE_CODE;
```

```
begin
```

```
    code_2' (FMOVE_D, X'REF, FP0);  
    code_1' (FATAN_D, FP0);  
    code_2' (FMOVE_D, FP0, RESULT'REF);  
end ATAN_68881;
```

```
procedure SQRT_68881 (X : in FLOAT; RESULT : out FLOAT) is
```

```
    use MACHINE_CODE;
```

```
begin
```

APPENDIX B
"VERDIX.SH" FILE

```
code_2' (FMOVE_D, X'REF, FP0);
code_1' (FSQRT_D, FP0);
code_2' (FMOVE_D, FP0, RESULT'REF);
end SQRT_68881;
```

procedure ETOX_68881(X : in FLOAT; RESULT : out FLOAT) is

use MACHINE_CODE;

```
begin
code_2' (FMOVE_D, X'REF, FP0);
code_1' (FETOX_D, FP0);
code_2' (FMOVE_D, FP0, RESULT'REF);
end ETOX_68881;
```

procedure LOG10_68881(X : in FLOAT; RESULT : out FLOAT) is

use MACHINE_CODE;

```
begin
code_2' (FMOVE_D, X'REF, FP0);
code_1' (FLOG10_D, FP0);
code_2' (FMOVE_D, FP0, RESULT'REF);
end LOG10_68881;
```

procedure LOGN_68881 (X : in FLOAT; RESULT : out FLOAT) is

use MACHINE_CODE;

```
begin
code_2' (FMOVE_D, X'REF, FP0);
code_1' (FLOGN_D, FP0);
code_2' (FMOVE_D, FP0, RESULT'REF);
end LOGN_68881;
```

```
pragma INLINE (SIN_68881, COS_68881, ATAN_68881, SQRT_68881, ETOX_68881);
pragma INLINE (LOG10_68881, LOGN_68881);
```

function SIN (X : FLOAT) return FLOAT is

RESULT : FLOAT;

```
begin
SIN_68881 (X, RESULT);
return RESULT;
end SIN;
```

APPENDIX B
"VERDIX.SH" FILE

function COS (X : FLOAT) return FLOAT is

RESULT : FLOAT;

begin

COS_68881 (X, RESULT);

return RESULT;

end COS;

function ATAN (X : FLOAT) return FLOAT is

RESULT : FLOAT;

begin

ATAN_68881 (X, RESULT);

return RESULT;

end ATAN;

function SQRT (X : FLOAT) return FLOAT is

RESULT : FLOAT;

begin

SQRT_68881 (X, RESULT);

return RESULT;

end SQRT;

function EXP (X : FLOAT) return FLOAT is

RESULT : FLOAT;

begin

ETOX_68881(X, RESULT);

return RESULT;

end EXP;

function LOG10 (X : FLOAT) return FLOAT is

RESULT : FLOAT;

begin

LOG10_68881 (X, RESULT);

return RESULT;

end LOG10;

APPENDIX B
"VERDIX.SH" FILE

function LOG(X : FLOAT) return FLOAT is

RESULT : FLOAT;

begin

LOGN_68881 (X, RESULT);

return RESULT;

end LOG;

begin

if FLOAT_SIZE /= FLOAT_SIZE then raise SIZE_ASSERTION_ERROR; end if;
end MATH;

--endmath

fi

cd \$cur_wd

fi

list:=[aA]000001.a* [aA]000021.a* [aA]000022.a*

[aA]000031.a* [aA]000032.a* [aA]000041.a* [aA]000042.a*

for x in \$list

do

y=`echo \$x | sed -n '1,\$s/.ada/.a/gp`

cp \$x \$pbd/sup/\$y

done

else

echo "sup directory already present - no files copied"

fi

if [\$choice = 1]

then

cat > \$pbd/sup/a000019.a << --endtime

--

-- This function returns the user time on a Unix system by interfacing to the c library to access -
-- the times routine

--

with system;

function CPU_TIME_CLOCK return DURATION is

-- ticks per second of the clock: from /usr/include/sys/param.h

HZ : constant := 100

-- time in ticks: from /usr/include/sys/types.h

type TIME_T is new INTEGER;

-- structure filled in by times(): from /usr/include/sys/times.h

type TMS_STRUCT is record

TMS_UTIME : TIME_T;

TMS_STIME : TIME_T;

TMS_CUTIME : TIME_T;

TMS_CSTIME : TIME_T;

APPENDIX B
"VERDIX.SH" FILE

```

end record;
TIME_BUF : TMS_STRUCT;
procedure TIMES (TIME_BUF: SYSTEM.ADDRESS);
pragma INTERFACE (C,TIMES);
begin
    TIMES (TIME_BUF'ADDRESS);
    return DURATION(DURATION (TIME_BUF.TMS_UTCIME)/HZ);
end CPU_TIME_CLOCK;
-- endtime
fi
if [$choice = 2 -o $choice = 3]
then
    cat > $pbd/sup/a000018.a << -- endtime
function CPU_TIME_CLOCK
    return DURATION;

with System;
package Unix_Resources is

type timeval is
    record
        sec : integer;
        usec : integer;
    end record; -- timeval

type rusage is
    record
        ru_utime : timeval; -- user mode time
        ru_stime : timeval; -- system mode time
        ru_vamrss : integer; -- max resident set (kbytes)
        ru_isrss : integer; -- memory shared with others (kbytes * seconds)
        ru_idrss : integer; -- unshared in memory (kbytes * seconds)
        ru_minflt : integer; -- page faults serviced w/o IO
        ru_majflt : integer; -- page faults serviced with IO
        ru_nswp : integer; -- number of times swapped out
        ru_inblock : integer; -- times of file system input
        ru_outblock : integer; -- times of file system output
        ru_msgsnd : integer; -- number of ipc messages sent
        ru_msgrcv : integer; -- number of ipc messages received
        ru_nsignals : integer; -- number of signals delivered
        ru_nvcsw : integer; -- number of times context switch due to process surrendering
        -- the CPU before timeout
        ru_nivcsw : integer; -- number of times context switch due to process timing out
        -- (or higher becomes eligible)
    end record; -- rusage

```

APPENDIX B
"VERDIX.SH" FILE

```

SELF : constant integer := 0;
CHILDREN : constant integer := -1;

procedure getrusage (who : integer;
                    rusage_buf : System.address);
pragma interface (C, getrusage);

end Unix_Resources;

with Unix_Resources;
function CPU_TIME_CLOCK
return DURATION
is
    rusage_buffer : Unix_Resources.rusage;
    problems : exception;
    seconds : float;
    useconds : float;
begin
    Unix_Resources.getrusage(Unix_Resources.SELF, rusage_buffer'address);
    seconds := float(rusage_buffer.ru_utime.sec + rusage_buffer.ru_stime.sec);
    useconds := float(rusage_buffer.ru_utime.usec + rusage_buffer.ru_stime.usec);
    return duration(seconds + (useconds / 1000000.0));
end CPU_TIME_CLOCK;

-- endtime
fi
if [$choice = 4]
then
    x={aA}000011.a*
    y='echo $x | sed -n '1,$s/.ada/.a/gp'
    cp $x $pbd/sup/$y
fi
if [-r A*.a*]
then
    list='A B C D E F G H L P T Y Z'
else
    list='a b c d e f g h l p t y z'
fi
echo "Copying tests and changing names (will take a while)..."
for x in $list
do
    if [! -d $pbd/$x]
    then
        a.mklib $pbd/$x $pbd/sup
        cd $pbd/$x
        # insert any extra a.path or a.info directives here, multiplexed
        # for release based on choice
    
```

APPENDIX B
"VERDIX.SH" FILE

```

# case $choice in
# 4) if [ $conf != 0 ]
# then
# a.path -i $conf_dir
# fi
# ;;
# *) ;;
# esac
cd $cur_wd
fi
if [ $x = a -o $x = A ]
then
    for y in [aA]00009[0-7].a*
    do
        z=`echo $y | sed -n '1,$s/.ada/.a/gp`
        cp $y $pbd/$x/$z
    done
    # go in and add the proper math package for test a000092
    sed -n -e 's/FLOAT_MATH_LIB/MATH/g' -e 'p' \
$pbdd/$x/[aA]000092.a > $pbdd/$x/ptmp
    mv $pbdd/$x/ptmp $pbdd/$x/a000092.a
else
    for y in $x*.a*
    do
        z=`echo $y | sed -n '1,$s/.ada/.a/gp`
        cp $y $pbd/$x/$z
    done
fi
done
echo "Done"
echo ""
echo "Installation is now complete. You are ready to run the tests. Re-run the script and"
echo "select the run option."
exit
fi
# This part of the script is for test modifications
if [ $option -eq 2 ]
then
    echo "PIWG does not endorse any test modifications. "
fi
# The script from here on down is used to run the PIWG suite
if [ -r $pbdd/sup/a000011.a -o -r $pbdd/sup/A000011.a ]
then
    cross=1
else
    cross=0

```


APPENDIX B
"VERDIX.SH" FILE

```
fi
echo ""
echo $n "Do you wish to run the tests WITHOUT suppressed checking (y or n)? $c"
read answer
if [ $answer = y -o $answer = Y ]
then
    ada_comp='-C ada -O9'
else
    ada_comp='-C ada -O9 -S'
fi
echo ""
echo "The support library contains all of the appropriate A files necessary to run all of the tests"
echo "(piwg_io, timing routine, etc). This needs to be compiled before any tests can be run."
echo ""
echo $n "Have you compiled the support library (y or n)? $c"
read answer
if [ $answer = n -o $answer = N ]
then
    cd $pbd/sup
    echo "Making Support Library..."
    a.make $ada_comp -v -f *.a
    cd $cur_wd
fi
echo "Enter desired report file (default is screen)"
echo $n "file> $c"
read rep_file
rep_pres='echo $rep_file | wc -w'
if [ $rep_pres != 0 ]
then
    foo='echo $rep_file | grep / | wc -w'
    if [ $foo = 0 ]
    then
        bar='pwd'
        foo=$bar/$rep_file
        rep_file=$foo
    fi
fi
echo $rep_file
while [ $TRUE ]
do
    echo ""
    echo " 1) Run all tests"
    echo " 2) Run tests by letter"
    echo " 3) Run individual tests"
    echo ""
    echo $n "Enter choice : $c"
```

APPENDIX B
"VERDIX.SH" FILE

```
read run_opt
case $run_opt in
  [1-3]) break;;
  *) echo " Valid options are 1-3";;
esac
done
run_list=""
if [ $run_opt = 3 ]
then
  echo "Enter desired test numbers separated by spaces. Multiple lines may be used."
  echo "Terminate with a blank line"
  while [ $TRUE ]
  do
    read one_line
    cnt=`echo $one_line | wc -w`
    if [ $cnt = 0 ]
    then
      break
    fi
    run_temp=`echo $run_list $one_line`
    run_list=$run_temp
  done
  for cur_test in $run_list
  do
    run_dir=`echo $cur_test | sed -n s/[0-9]//gp`
    cd $pbd/$run_dir
    test_name=$cur_test
    if [ ! -r $test_name.a ]
    then
      test_name=`echo $cur_test | sed -n s/$/m/gp`
    fi
    test_files=`echo $cur_test | sed -n s/$/^*.a/gp`
    echo "Building test $test_name..."
    a.make $ada_comp -v -f $test_files
    if [ $cross = 0 ]
    then
      chk_math=`grep 'with *MATH' $test_files | wc -w`
      if [ $chk_math -ge 1 ]
      then
        a.make $ada_comp -v $test_name -lm -o $test_name.out
      else
        a.make $ada_comp -v $test_name -o $test_name.out
      fi
    fi
    if [ ! -s $test_name.out ]
    then
```

APPENDIX B
"VERDIX.SH" FILE

```

        echo "Could not create test $cur_test"
    else
        echo "Executing $test_name"
        then
            if [$rep_pres = 0]
            then
                a.run $test_name.out
            else
                a.run $test_name.out >> $rep_file 2>&1
            fi
        else
            if [$rep_pres = 0]
            then
                $test_name.out
            else
                $test_name.out >> $rep_file 2>&1
            fi
        fi
        cd $cur_wd # kludgey, but the easiest way around relative paths
    done
    exit
fi
if [$run_opt = 2]
then
    echo "Enter desired test letters separated by spaces. Multiple lines may be used."
    echo "Terminate with a blank line"
    while [ $TRUE ]
    do
        read one_line
        cnt=`echo $one_line | wc -w`
        if [$cnt = 0]
        then
            break
        fi
        run_temp=`echo $run_list $one_line`
        run_list=$run_temp
    done
fi
if [$run_opt = 1]
then
    run_list='a b c d e f g h i p t y'
fi
for cur_lib in $run_list
do

```

APPENDIX B
"VERDIX.SH" FILE

```

cd $pbd/$cur_lib
echo "Bringing units up to date..."
a.make $ada_comp -v -f *.a
exec_t1=$cur_lib*m.a
foo=`echo $exec_t1`
srch=`echo "$foo" | sed -n 's/^*//gp' | wc -w`
if [$srch != 0]
then
    exec_t1=""
fi
exec_t2=$cur_lib*[0-9].a
foo=`echo $exec_t2`
srch=`echo "$foo" | sed -n 's/^*//gp' | wc -w`
if [$srch != 0]
then
    exec_t2=""
fi
exec_list=`echo "$exec_t1 $exec_t2"`
for cur_unit in $exec_list
do
    cur_test=`echo $cur_unit | sed -n 's/\.a//gp'`
    echo "Building test $cur_test"
    if [$cross = 0]
    then
        chk_math=`grep 'with *MATH' $cur_unit | wc -w`
        if [$chk_math -ge 1]
        then
            a.make $ada_comp -v $cur_test -o $cur_test.out -lm
        else
            a.make $ada_comp -v $cur_test -o $cur_test.out
        fi
    fi
    if [! -s $cur_test.out]
    then
        echo "Could not create test $cur_test"
    else
        echo "Executing $cur_test"
        if [$cross = 1]
        then
            if [$rep_pres = 0]
            then
                a.run $cur_test.out
            else
                a.run $cur_test.out >> $rep_file 2>&1
            fi
        else

```

APPENDIX B
"VERDIX.SH" FILE

```
if [$rep_pres = 0]
then
    Scur_test.out
else
    Scur_test.out >> $rep_file 2>&1
fi
fi
done
cd $cur_wd
done
```

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: A000090

Clock resolution measurement running.

Test Description: Determine clock resolution using second differences of values returned by the function CPU_Time_Clock.

Number of sample values is 12000
Clock Resolution = 0.0100000000000000 seconds.
Clock Resolution (average) = 0.0100000000000000 seconds.
Clock Resolution (variance) = 0.0000000000000000 seconds.

Capture a few sizes for later analysis

8 Boolean's size
8 Boolean_object's size
8 Character's size
8 Character_object's size
32 Positive's size
32 Positive_object's size
32 Integer's size
64 Float's size
32 Duration's size
0.0001 Duration's small
0.0100 System.Tick
0.0202 = clock resolution used for iteration stability

Test Name: A000091 Class Name: Composite
CPU Time: 33.50 MICROSECONDS plus or minus 1.675
Wall/CPU: 1.00 ratio Iteration Count: 204800
Test Description: Reinhold P. Weicker's DHRYSTONE composite benchmark.

0.0202 = clock resolution used for iteration stability

Test Name: A000092 Class Name: Composite
CPU Time: 106.50 MILLISECONDS plus or minus 5.325
Wall/CPU: 1.01 ratio Iteration Count: 40
Test Description: Ada version of the Whetstone Benchmark Program. Manufacturers math routines.

Average Whetstone rating: 9390 KWIPS
0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: A000093 Class Name: Composite
CPU Time: 100.13 MILLISECONDS plus or minus 5.006
Wall/CPU: 1.01 ratio Iteration Count: 80
Test Description: Ada version of the Whetstone Benchmark Program. Built in 'standard' math routines.

Average Whetstone rating: 9988 KWIPS

Test Name: A000094 Class: Composite
Perm 0.10
Towers 0.16
Queens 0.05
Intrmm 0.09
Mm 0.04
Puzzle 0.40
Quick 0.08
Bubble 0.16
Tree 0.15
FFT 0.18
Ack 8.18

Test Description: Henessy benchmarks

A000095 collects information interactively, then reads the output from COMPILE.COM or other script run and produces a compressed file of results.

Enter file name of input file:

**** MAIN PROGRAM ABANDONED -- EXCEPTION "NAME_ERROR" RAISED**

0.0202 = clock resolution used for iteration stability

Test Name: B000010 Class Name: Composite
CPU Time: 180.75 MILLISECONDS plus or minus 9.038
Wall/CPU: 1.00 ratio Iteration Count: 40
Test Description: NASA Orbit determination heavy floating point and trig initializing and running.

4.71471409E+03 4.67389156E+03 2.38594821E+02

-5.33895085E+00 5.19603808E+00 2.25675781E+00

0.0202 = clock resolution used for iteration stability

APPENDIX C

FWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: B000013	Class Name: Composite
CPU Time: 2.34 MILLISECONDS	plus or minus 0.117
Wall/CPU: 1.01 ratio	Iteration Count: 3200
Test Description: TRACKER CENTROID Algorithm. All integer calculations searching a 60 x 60 array.	

0.0202 = clock resolution used for iteration stability

WALL time less than CPU time

Test Name: C000001	Class Name: Tasking
CPU Time: 389.06 MICROSECONDS	plus or minus 19.453
Wall/CPU: 1.00 ratio	Iteration Count: 12800
Test Description: Task create and terminate measurement with one task, no entries, when task is in a procedure using a task type in a package, no select statement, no loop,	

0.0202 = clock resolution used for iteration stability

Test Name: C000002	Class Name: Tasking
CPU Time: 374.22 MICROSECONDS	plus or minus 18.711
Wall/CPU: 1.00 ratio	Iteration Count: 12800
Test Description: Task create and terminate time measurement with one task, no entries when task is in a procedure, task defined and used in procedure, no select statement, no loop.	

0.0202 = clock resolution used for iteration stability

Test Name: C000003	Class Name: Tasking
CPU Time: 361.72 MICROSECONDS	plus or minus 18.086
Wall/CPU: 1.00 ratio	Iteration Count: 12800
Test Description: Task create and terminate time measurement. Task is in declare block of main procedure one task, no entries, task is in the loop.	

0.0202 = clock resolution used for iteration stability

Test Name: D:\00001	Class Name: Allocation
CPU Time: 1.39 MICROSECONDS	plus or minus 0.070
Wall/CPU: 1.00 ratio	Iteration Count: 1638400
Test Description: Dynamic array allocation, use and deallocation time measurement. Dynamic array elaboration, 1000 integers in a procedure get space and free it in the procedure on each call.	

0.0202 = clock resolution used for iteration stability

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

0.0202 = clock resolution used for iteration stability

0.0202 = clock resolution used for iteration stability

0.0202 = clock resolution used for iteration stability

0.0202 = clock resolution used for iteration stability.

0.0202 = clock resolution used for iteration stability.

APPENDIX C

Test Name: E000003	Class Name: Exception
CPU Time: 35.50 MICROSECONDS	plus or minus 1.775
Wall/CPU: 1.01 ratio	Iteration Count: 204800
Test Description: Exception raise and handle timing measurement when exception is raised nested 3 deep in procedure calls.	

0.0202 = clock resolution used for iteration stability

Test Name: E000004	Class Name: Exception
CPU Time: 46.29 MICROSECONDS	plus or minus 2.315
Wall/CPU: 1.01 ratio	Iteration Count: 102400
Test Description: Exception raise and handle timing measurement when exception is nested 4 deep in procedures.	

0.0202 = clock resolution used for iteration stability

Test Name: E000005	Class Name: Exception
CPU Time: 42.58 MICROSECONDS	plus or minus 3.156
Wall/CPU: 1.00 ratio	Iteration Count: 25600
Test Description: Exception raise and handle timing measurement when exception is in a rendezvous. Both the task and the caller must handle the exception.	

0.0202 = clock resolution used for iteration stability

***** POSSIBLY INACCURATE MEASUREMENT *****

Test Name: F000001	Class Name: Style
CPU Time: 0.09 MICROSECONDS	plus or minus 0.025
Wall/CPU: 1.00 ratio	Iteration Count: 3276800
Test Description: Time to set a boolean flag using a logical equation, a local, and a global integer are compared. Compare this test with F000002	

0.0202 = clock resolution used for iteration stability

***** POSSIBLY INACCURATE MEASUREMENT *****

Test Name:	F000002	Class Name:	Style
CPU Time:	0.10 MICROSECONDS	plus or minus	0.025
Wall/CPU:	1.01 ratio	Iteration Count:	3276800
Test Description:	Time to set a boolean flag using an 'if' test, a local, and a global integer are compared. Compare this test with F000001.		

0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: G000001 Class Name: Input/Output
CPU Time: 39.06 MICROSECONDS plus or minus 1.973
Wall/CPU: 1.00 ratio Iteration Count: 40960
Test Description: TEXT_IO.GET_LINE reading 20 characters, time measured. A scratch file is written, then read and reset.

0.0202 = clock resolution used for iteration stability

Test Name: G000002 Class Name: Input/Output
CPU Time: 51.51 MICROSECONDS plus or minus 2.576
Wall/CPU: 1.02 ratio Iteration Count: 40960
Test Description: TEXT_IO.GET called 20 times per line, time measured. A scratch file is written, then read and reset. Compare to G000001 for about same number of characters.

0.0202 = clock resolution used for iteration stability

Test Name: G000003 Class Name: Input/Output
CPU Time: 196.29 MICROSECONDS plus or minus 9.815
Wall/CPU: 1.02 ratio Iteration Count: 20480
Test Description: TEXT_IO.PUT_LINE for 20 characters, timing measurement. A scratch file is opened, written and reset.

0.0202 = clock resolution used for iteration stability

Test Name: G000004 Class Name: Input/Output
CPU Time: 44.43 MICROSECONDS plus or minus 2.222
Wall/CPU: 1.00 ratio Iteration Count: 40960
Test Description: TEXT_IO.PUT 20 times with one character, time measurement. A scratch file is written, reset and rewritten. Compare, approximately, to G000003.

0.0202 = clock resolution used for iteration stability

Test Name: G000005 Class Name: Input/Output
CPU Time: 19.04 MICROSECONDS plus or minus 0.952
Wall/CPU: 1.00 ratio Iteration Count: 163840
Test Description: TEXT_IO.GET an integer from a local string, timing measurement. Use TEXT_IO.PUT to convert 1..100 to a string, then use TEXT_IO.GET to get the number back.

0.0202 = clock resolution used for iteration stability

APPENDIX C

FWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: G000006 Class Name: Input/Output
CPU Time: 122.07 MICROSECONDS plus or minus 6.104
Wall/CPU: 1.01 ratio Iteration Count: 20480
Test Description: TEXT_IO.GET getting a floating point fraction from a local string. Timing measurement on 0.001 to 0.01 range of numbers. Compare, approximately, to G000005 for integer vs float.

0.0202 = clock resolution used for iteration stability

Test Name: G000007 Class Name: Input/Output
CPU Time: 1031.25 MICROSECONDS plus or minus 51.563
Wall/CPU: 1.00 ratio Iteration Count: 5120
Test Description: Open and close an existing file, time measurement. A scratch file is created and closed. The scratch file is opened IN_FILE and closed in a loop

0.0202 = clock resolution used for iteration stability

Test Name: H000001 Class Name: Chapter 13
CPU Time: 1.21 MICROSECONDS plus or minus 0.061
Wall/CPU: 1.01 ratio Iteration Count: 3276800
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are PACKED with the pragma 'PACK.' For this test the operations are performed on the entire arrays.

0.0202 = clock resolution used for iteration stability

Test Name: H000002 Class Name: Chapter 13
CPU Time: 18.29 MICROSECONDS plus or minus 0.914
Wall/CPU: 1.00 ratio Iteration Count: 409600
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are NOT PACKED with the pragma 'PACK.' For this test the operations are performed on the entire array.

0.0202 = clock resolution used for iteration stability

Test Name: H000003 Class Name: Chapter 13
CPU Time: 44.82 MICROSECONDS plus or minus 2.241
Wall/CPU: 1.00 ratio Iteration Count: 102400
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are PACKED with the pragma 'PACK.' For this test the operations are performed on components in a loop

0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: H000004 Class Name: Chapter 13
CPU Time: 18.48 MICROSECONDS plus or minus 0.924
Wall/CPU: 1.01 ratio Iteration Count: 409600
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are NOT PACKED with the pragma 'PACK.' For this test the operations are performed on components in a loop.

0.0202 = clock resolution used for iteration stability

***** POSSIBLY INACCURATE MEASUREMENT *****

Test Name: H000005 Class Name: Chapter 13
CPU Time: 0.00 MICROSECONDS plus or minus 0.025
Wall/CPU: 1.00 ratio Iteration Count: 3276800
Test Description: The time for UNCHECKED_CONVERSION to move one INTEGER object to another INTEGER object. This may be zero with good optimization.

0.0202 = clock resolution used for iteration stability

Test Name: H000006 Class Name: Chapter 13
CPU Time: 4.27 MICROSECONDS plus or minus 0.214
Wall/CPU: 1.00 ratio Iteration Count: 819200
Test Description: The time for UNCHECKED_CONVERSION to move 10 floating array objects to a 10 component floating record.

0.0202 = clock resolution used for iteration stability

Test Name: H000007 Class Name: Chapter 13
CPU Time: 3.13 MICROSECONDS plus or minus 0.156
Wall/CPU: 1.00 ratio Iteration Count: 1638400
Test Description: The time to store and extract bit fields that are defined by representation clauses using Boolean and Integer record components. 12 accesses, 5 stores, 1 record copy.

0.0202 = clock resolution used for iteration stability

Test Name: H000009 Class Name: Chapter 13
CPU Time: 4.79 MICROSECONDS plus or minus 0.239
Wall/CPU: 1.02 ratio Iteration Count: 819200
Test Description: The time to perform a change of representation. If the result is near zero, feature probably not implemented.

0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: L000001 Class Name: Iteration
 CPU Time: 0.13 MICROSECONDS plus or minus 0.008
 Wall/CPU: 1.01 ratio Iteration Count: 10240000
 Test Description: Simple "for" loop time for I in 1 .. 100 loop time reported is for once through loop.

0.0202 = clock resolution used for iteration stability

Test Name: L000002 Class Name: Iteration
 CPU Time: 0.13 MICROSECONDS plus or minus 0.008
 Wall/CPU: 1.00 ratio Iteration Count: 10240000
 Test Description: Simple "while" loop time while I <= 100 loop time reported is for once through loop.

0.0202 = clock resolution used for iteration stability

Test Name: L000003 Class Name: Iteration
 CPU Time: 0.15 MICROSECONDS plus or minus 0.008
 Wall/CPU: 1.00 ratio Iteration Count: 10240000
 Test Description: Simple "exit" loop time, loop I:=I+1; exit when I>100; end loop; time reported is for once through loop.

0.0202 = clock resolution used for iteration stability

Test Name: L000004 Class Name: Iteration
 CPU Time: 0.12 MICROSECONDS plus or minus 0.010
 Wall/CPU: 1.00 ratio Iteration Count: 8192000
 Test Description: Measures Compiler's choice to UNWRAP a small loop of five (5) iterations when given a PRAGMA OPTIMIZE(TIME). An execution time less than .05 microseconds indicates the unwrap occurred.

0.0202 = clock resolution used for iteration stability

Test Name: L000005 Class Name: Iteration
 CPU Time: 0.12 MICROSECONDS plus or minus 0.010
 Wall/CPU: 1.01 ratio Iteration Count: 8192000
 Test Description: Measures Compiler's choice to UNWRAP a small loop of five (5) iterations when given a PRAGMA OPTIMIZE (Space). An execution speed < .05 microseconds indicates the unwrap occurred.

0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: P000001 Class Name: Procedure
CPU Time: 0.28 MICROSECONDS plus or minus 0.014
Wall/CPU: 1.01 ratio Iteration Count: 8192000
Test Description: Procedure call and return time (may be zero if automatic inlining) procedure is local, no parameters.

0.0202 = clock resolution used for iteration stability

Test Name: P000002 Class Name: Procedure
CPU Time: 0.28 MICROSECONDS plus or minus 0.014
Wall/CPU: 1.01 ratio Iteration Count: 8192000
Test Description: Procedure call and return time. Procedure is local, no parameters when procedure is not inlinable.

0.0202 = clock resolution used for iteration stability

Test Name: P000003 Class Name: Procedure
CPU Time: 0.28 MICROSECONDS plus or minus 0.014
Wall/CPU: 1.01 ratio Iteration Count: 8192000
Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. Compare to P000002

0.0202 = clock resolution used for iteration stability

Test Name: P000005 Class Name: Procedure
CPU Time: 0.38 MICROSECONDS plus or minus 0.020
Wall/CPU: 1.01 ratio Iteration Count: 4096000
Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. One parameter, in INTEGER

0.0202 = clock resolution used for iteration stability

Test Name: P000006 Class Name: Procedure
CPU Time: 0.33 MICROSECONDS plus or minus 0.020
Wall/CPU: 1.00 ratio Iteration Count: 4096000
Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. One parameter, out INTEGER.

0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: P000007 Class Name: Procedure
CPU Time: 0.42 MICROSECONDS plus or minus 0.021
Wall/CPU: 1.01 ratio Iteration Count: 4096000
Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. One parameter, in out INTEGER

0.0202 = clock resolution used for iteration stability

Test Name: P000010 Class Name: Procedure
CPU Time: 1.86 MICROSECONDS plus or minus 0.093
Wall/CPU: 1.00 ratio Iteration Count: 2048000
Test Description: Procedure call and return time measurement. Compare to P000005
10 parameters, in INTEGER

0.0202 = clock resolution used for iteration stability

Test Name: P000011 Class Name: Procedure
CPU Time: 3.90 MICROSECONDS plus or minus 0.195
Wall/CPU: 1.00 ratio Iteration Count: 1024000
Test Description: Procedure call and return time measurement. Compare to P000005, P000010
20 parameters, in INTEGER.

0.0202 = clock resolution used for iteration stability

Test Name: P000012 Class Name: Procedure
CPU Time: 1.79 MICROSECONDS plus or minus 0.089
Wall/CPU: 1.00 ratio Iteration Count: 2048000
Test Description: Procedure call and return time measurement. Compare with P000010
(discrete vs composite parameters) 10 paramaters, in MY_RECORD a three component record.

0.0202 = clock resolution used for iteration stability

Test Name: P000013 Class Name: Procedure
CPU Time: 3.71 MICROSECONDS plus or minus 0.186
Wall/CPU: 1.00 ratio Iteration Count: 1024000
Test Description: Procedure call and return time measurement 20 composite 'in' parameters
the composite type is a 3 component record.

0.0202 = clock resolution used for iteration stability

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

Test Name: T000001 Class Name: Tasking
CPU Time: 143.16 MICROSECONDS plus or minus 7.158
Wall/CPU: 1.01 ratio Iteration Count: 51200
Test Description: Minimum rendezvous, entry call and return time 1 task 1 entry, task inside procedure no select.

0.0202 = clock resolution used for iteration stability

Test Name: T000002 Class Name: Tasking
CPU Time: 136.52 MICROSECONDS plus or minus 6.826
Wall/CPU: 1.00 ratio Iteration Count: 51200
Test Description: Task entry call and return time measured. One task active, one entry in task, task in a package, no select statement.

0.0202 = clock resolution used for iteration stability

Test Name: T000003 Class Name: Tasking
CPU Time: 136.33 MICROSECONDS plus or minus 6.816
Wall/CPU: 1.01 ratio Iteration Count: 51200
Test Description: Task entry call and return time measured. Two tasks active, one entry per task, tasks in a package, no select statement.

0.0202 = clock resolution used for iteration stability

Test Name: T000004 Class Name: Tasking
CPU Time: 143.16 MICROSECONDS plus or minus 7.158
Wall/CPU: 1.00 ratio Iteration Count: 51200
Test Description: Task entry call and return time measured. One tasks active, two entries, tasks in a package using select statement.

0.0202 = clock resolution used for iteration stability

Test Name: T000005 Class Name: Tasking
CPU Time: 146.88 MICROSECONDS plus or minus 7.344
Wall/CPU: 1.02 ratio Iteration Count: 32000
Test Description: Task entry call and return time measured. Ten tasks active, one entry per task, tasks in a package, no select statement.

0.0202 = clock resolution used for iteration stability

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

0.0202 = clock resolution used for iteration stability

0.0202 = clock resolution used for iteration stability

Test Name: Y000001 Measure actual delay vs commanded delay

Y000002 Measure CPU_TIME_CLOCK against a watch for 1 minute. See if every 5 seconds are ticked off about right.

53

APPENDIX C

PIWG (FIRST) EXECUTION TESTS RESULT USING SPARCADA ON THE SUN WORKSTATION

30
35
40
45
50
55
60
65

Y000003 Measure CALENDAR.CLOCK against a watch for 1 minute. See if every 5 seconds are ticked off about right.

0

** MAIN PROGRAM ABANDONED -- EXCEPTION "storage_error" RAISED

Error in kernel:: exception_handler: below bottom of user stack.

APPENDIX D

**SCRIPT FILE FOR
"SECOND RUN" PIWG TESTS
ON THE SUN WORKSTATION**

APPENDIX D
SCRIPT FILE FOR "SECOND RUN" PIWG TESTS ON THE SUN WORKSTATION

```
# File SECOND
# This is a script file to perform the SECOND RUN benchmark as described in the 8_1_90piwg
# READ.ME file for the UNIX VERDIX version of PIWG running on the SUN station. This
# version of SECOND was intended to reside and execute in the subdirectory Z where all the
# "Z" files, "Z000001.A" thru "Z000023.A" also reside.
#
echo ' compiling files separately '
echo ' start time '
/home/corbeaux/davidson/8_1_90piwg/sup/a000051.out
ada z000001.a
ada z000002.a
ada z000003.a
ada z000004.a
ada z000005.a
ada z000006.a
ada z000007.a
ada z000008.a
ada z000009.a
ada z000010.a
ada z000011.a
ada z000012.a
ada z000013.a
ada z000014.a
ada z000015.a
ada z000016.a
ada z000016a.a
ada z000017.a
ada z000017a.a
ada -M z000018.a -o z000018.out
z000018.out
ada z000020.a
ada z000021.a
ada z000022.a
ada -M z000023.a -o z000023.out
z000023.out
/home/corbeaux/davidson/8_1_90piwg/sup/a000051.out
echo ' end time '
echo ' '
echo 'Difference between the Start and Stop Wall Times represents a composite software'
echo 'development benchmark if no other tasks are running on the machine under test.'
```

APPENDIX E

**PIWG "SECOND RUN" RESULTS
USING
SPARCADA ON THE SUN WORKSTATION**

APPENDIX E
PIWG "SECOND RUN" RESULTS USING SPARCADA ON THE SUN WORKSTATION

compiling files separately

start time

CPU time now= 0.0400 WALL time now= 64069.1366 seconds.

Test printout and value of acceleration, 9.80665000000000E+00 meter per second squared = G

1.10324812500000E+01 meter

1.50000000000014E+00 second

2.08030461473005E+01 meter per second

UP SORTED DATA

1	1.00000000000000E+00	AAA FIRST	1.10
2	2.00000000000000E+00	BBB SECOND	2.10
3	3.00000000000000E+00	CCC THIRD	3.10
4	4.00000000000000E+00	DDD FOURTH	4.10

DOWN SORTED DATA

4	4.00000000000000E+00	DDD FOURTH	4.10
3	3.00000000000000E+00	CCC THIRD	3.10
2	2.00000000000000E+00	BBB SECOND	2.10
1	1.00000000000000E+00	AAA FIRST	1.10

in the bag

gone fishing

end FISH

ALL_STATEMENTS_PROCEDURE_2

into LOOP_NAME_1

Z000021 finished

CPU time now= 0.0100 WALL time now= 64395.3406 seconds.

end time

Difference between the Start and Stop Wall Times represents a composite software development benchmark if no other tasks are running on the machine under test.

APPENDIX F

**SCRIPT FILE FOR
"THIRD RUN" PIWG TESTS
ON THE SUN WORKSTATION**

APPENDIX F
SCRIPT FILE FOR "THIRD RUN" PIWG TESTS ON THE SUN WORKSTATION

```
# File THIRD
# This is a script file to perform the THIRD RUN benchmark as described in the 8_1_90piwg
# READ.ME file for the UNIX VERDIX version of PIWG running on the SUN station. This
# version of THIRD was intended to reside and execute in the subdirectory ZOTHER where all
# the "Z" files, "Z000100.A" thru "Z000200.A" also reside.
#
echo 'compiling files separately'
echo 'start time'
/home/corbeaux/davidson/8_1_90piwg/sup/a000051.out
ada z000100.a
ada z000101.a
ada z000102.a
ada z000103.a
ada z000104.a
ada z000105.a
ada z000106.a
ada z000107.a
ada z000108.a
ada z000109.a
ada z000110.a
ada z000111.a
ada z000112.a
ada z000113.a
ada z000114.a
ada z000115.a
ada z000116.a
ada z000117.a
ada z000118.a
ada z000119.a
ada z000120.a
ada z000121.a
ada z000122.a
ada z000123.a
ada z000124.a
ada z000125.a
ada z000126.a
ada z000127.a
ada z000128.a
ada z000129.a
ada z000130.a
ada z000131.a
ada z000132.a
ada z000133.a
ada z000134.a
ada z000135.a
ada z000136.a
ada z000137.a
ada z000138.a
ada z000139.a
ada z000140.a
ada z000141.a
ada z000142.a
ada z000143.a
ada z000144.a
```

APPENDIX F
SCRIPT FILE FOR "THIRD RUN" PIWG TESTS ON THE SUN WORKSTATION

ada z000145.a
ada z000146.a
ada z000147.a
ada z000148.a
ada z000149.a
ada z000150.a
ada z000151.a
ada z000152.a
ada z000153.a
ada z000154.a
ada z000155.a
ada z000156.a
ada z000157.a
ada z000158.a
ada z000159.a
ada z000160.a
ada z000161.a
ada z000162.a
ada z000163.a
ada z000164.a
ada z000165.a
ada z000166.a
ada z000167.a
ada z000168.a
ada z000169.a
ada z000170.a
ada z000171.a
ada z000172.a
ada z000173.a
ada z000174.a
ada z000175.a
ada z000176.a
ada z000177.a
ada z000178.a
ada z000179.a
ada z000180.a
ada z000181.a
ada z000182.a
ada z000183.a
ada z000184.a
ada z000185.a
ada z000186.a
ada z000187.a
ada z000188.a
ada z000189.a
ada z000190.a
ada z000191.a
ada z000192.a
ada z000193.a
ada z000194.a
ada z000195.a
ada z000196.a
ada z000197.a
ada z000198.a

APPENDIX F
SCRIPT FILE FOR "THIRD RUN" PIWG TESTS ON THE SUN WORKSTATION

```
ada z000199.a
/home/corbeaux/davidson/8_1_90piwg/sup/a000051.out
echo 'end time'
echo ' '
echo ' '
echo 'compiling files at once in z000200.a'
echo 'start time'
/home/corbeaux/davidson/8_1_90piwg/sup/a000051.out
ada z000200.a
/home/corbeaux/davidson/8_1_90piwg/sup/a000051.out
echo 'end time'
```

APPENDIX G

**PIWG "THIRD RUN" RESULTS
USING
SPARCADA ON THE SUN WORKSTATION**

APPENDIX G
PIWG "THIRD RUN" RESULTS USING SPARCADA ON THE SUN WORKSTATION

compiling files separately

start time

CPU time now= 0.0300 WALL time now= 37158.7212 seconds.

CPU time now= 0.0200 WALL time now= 37680.7195 seconds.

end time

compiling files at once in z000200.a

start time

CPU time now= 0.0200 WALL time now= 37680.7587 seconds.

CPU time now= 0.0200 WALL time now= 38496.1001 seconds.

end time

APPENDIX H

"COMPILE.COM" FILE

APPENDIX H
"COMPILE.COM" FILE

```
$! RUN THIS FIRST ON VAX VMS (Feature benchmarks)
$!
$! VAX VMS files to compile (using standard output PIWG_IO)
$ SET VERIFY
$ SET NOON
$ SET DEF $!$DIA?:[DAVIDSON.PIWG]
$ DEL [.ADALIB]*.*;* ! Clean out any old libraries
$ ACS CREATE LIB [.ADALIB]
$ ACS SET LIB [.ADALIB]
$ @ VAXCONFIG
$ ADA A000001
$ ADA A000012 !VAX Ada dependent
$ ADA A000021
$ ADA A000022
$ ADA A000031
$ ADA A000032
$ ADA A000041
$ ADA A000042
$ ADA A000047 ! PIWG_TIMER_GENERIC spec
$ ADA A000048 ! PIWG_TIMER_GENERIC body
$ ADA A000051 !wall timing routines for host only
$ ADA A000052 !
$ ADA A000053 !
$ ADA A000054 !
$ ADA A000055 !
$ ADA A000090
$ ADA A000091
$ ADA A000092 !VAX Ada dependent
$ ADA A000093
$ ADA A000094 !just for comparison, now split up into A..K
$ ADA A000094A
$ ADA A000094B
$ ADA A000094C
$ ADA A000094D
$ ADA A000094E
$ ADA A000094F
$ ADA A000094G
$ ADA A000094H
$ ADA A000094I
$ ADA A000094J
$ ADA A000094K
$ ADA A000095 !program to process output file
$ ADA B000001A
$ ADA B000001B/NOCHECK/WARN=(WARN:NONE) !make equivalent of pragma suppress
$ ADA B000002A
$ ADA B000002B/NOCHECK/WARN=(WARN:NONE)
$ ADA B000003A
$ ADA B000003B/NOCHECK/WARN=(WARN:NONE)
$ ADA B000004A
$ ADA B000004B/NOCHECK/WARN=(WARN:NONE)
$ ADA B000010
$ ADA B000011
$ ADA B000013
$ ADA C000001
```


APPENDIX H
"COMPILE.COM" FILE

\$ ADA C000002
\$ ADA C000003
\$ ADA D000001
\$ ADA D000002
\$ ADA D000003
\$ ADA D000004
\$ ADA E000001
\$ ADA E000002
\$ ADA E000003
\$ ADA E000004
\$ ADA E000005
\$ ADA F000001
\$ ADA F000002
\$ ADA G000001
\$ ADA G000002
\$ ADA G000003
\$ ADA G000004
\$ ADA G000005
\$ ADA G000006
\$ ADA G000007
\$ ADA H000001
\$ ADA H000002
\$ ADA H000003
\$ ADA H000004
\$ ADA H000005
\$ ADA H000006
\$ ADA H000007
\$ ADA H000008
\$ ADA H000009
\$ ADA L000001
\$ ADA L000002
\$ ADA L000003
\$ ADA L000004
\$ ADA L000005
\$ ADA P000001
\$ ADA P000002
\$ ADA P000003
\$ ADA P000004
\$ ADA P000005
\$ ADA P000006
\$ ADA P000007
\$ ADA P000010
\$ ADA P000011
\$ ADA P000012
\$ ADA P000013
\$ ADA T000001
\$ ADA T000002
\$ ADA T000003
\$ ADA T000004
\$ ADA T000005
\$ ADA T000006
\$ ADA T000007 !execution possibly machine dependent
\$ ADA T000008

APPENDIX H
"COMPILE.COM" FILE

```
$ ADA Y000001
$ ADA Y000002
$ ADA Y000003
$ ADA A000100
$ ACS LINK A000100
$ RUN A000100 !all feature tests in one procedure
$!                !run one feature at a time if results are better
$!                ! mail in printout
$ ADA A000101 !about half the features
$ ADA A000102 !the other half
$ ACS LINK A000101
$ ACS LINK A000102
$ ACS LINK A000051
$ ACS LINK A000052
$ ACS LINK A000053
$ ACS LINK A000054
$ ACS LINK A000055
$ ACS LINK A000095
$ ACS LINK Y000002
$ ACS LINK Y000003
$ RUN A000051
$ RUN A000052
$ RUN A000053
$ RUN A000054
$ RUN A000101
$ RUN A000102
$ RUN A000055
$ DEL SCRATCH*.*;* !remove scratch files from "G" tests
$ DEL A000052D.*;* !remove intermediate timing files
$ DEL PIWGRES.*;* !remove saved results if any
$! DEL [.ADALIB]*.*;* !remove compilation library
```

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS

USING

DEC ADA ON THE VAX COMPUTER

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

```
$!SYLOGIN.COM
$!For the AAAF CLuster
$ SET NOVERIFY
$ SET NOON
$ SET DEF $!$DIA9:[DAVIDSON.PIWG]
$ DEL [.ADALIB]*.*;*    !Clean out any old libraries
$ ACS CREATE LIB [.ADALIB]
%ACS-I-CL_LIBCRE, Library $!$DIA9:[DAVIDSON.PIWG.ADALIB] created
$ ACS SET LIB [.ADALIB]
%ACS-I-CL_LIBIS, Current program library is
$!$DIA9:[DAVIDSON.PIWG.ADALIB]
$ @ VAXCONFIG
  System configuration:
    Cluster:  AAAF
    Node:     KNIGHT
    CPU:      VAX 4000-200
      Character emulated: TRUE
      F_FLOAT emulated:  FALSE
      D_FLOAT emulated:  FALSE
      G_FLOAT emulated:  FALSE
      H_FLOAT emulated:  TRUE
    VMS version: V5.5
    Virtual page count: 240000
    Working set maximum: 16400
    Main Memory (32.00Mb)
    System device: RF72
    User device:  RF71
  VAX Ada software configuration:
    ADA version: "VAX Ada V2.2-38"
    ACS version: "VAX Ada V2.2-38"
    ADARTL version: "V5.4-03"
    ADAMSG version: "1-029"
  Process configuration:
    Open file limit:  50
    Enqueue quota:    100
    Timer queue quota: 20
    Page file quota:  17000
    Working set memory parameters:
      WSQUOTA:  2500
      WSEXTENT: 4000
      Adjustment: enabled
$ EXIT          !0
$ ADA A000001
$ ADA A000012 !VAX Ada dependent
$ ADA A000021
$ ADA A000022
```

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

\$ ADA A000031
\$ ADA A000032
\$ ADA A000041
\$ ADA A000042
\$ ADA A000047 !PIWG_TIMER_GENERIC spec
\$ ADA A000048 !PIWG_TIMER_GENERIC body
\$ ADA A000051 !wall timing routines for host only
\$ ADA A000052 !
\$ ADA A000053 !
\$ ADA A000054 !
\$ ADA A000055 !
\$ ADA A000090
\$ ADA A000091
\$ ADA A000092 !VAX Ada dependent
\$ ADA A000093
\$ ADA A000094 !just for comparison, now split up into A..K
\$ ADA A000094A
\$ ADA A000094B
\$ ADA A000094C
\$ ADA A000094D
\$ ADA A000094E
\$ ADA A000094F
\$ ADA A000094G
\$ ADA A000094H
\$ ADA A000094I
\$ ADA A000094J
\$ ADA A000094K
\$ ADA A000095 !program to process output file
\$ ADA B000001A
\$ ADA B000001B/NOCHECK/WARN=(WARN:NONE) !make equivalent of pragma suppress
\$ ADA B000002A
\$ ADA B000002B/NOCHECK/WARN=(WARN:NONE)
\$ ADA B000003A
\$ ADA B000003B/NOCHECK/WARN=(WARN:NONE)
\$ ADA B000004A
\$ ADA B000004B/NOCHECK/WARN=(WARN:NONE)
\$ ADA B000010
\$ ADA B000011
\$ ADA B000013
\$ ADA C000001
\$ ADA C000002
\$ ADA C000003
\$ ADA D000001
\$ ADA D000002
\$ ADA D000003
\$ ADA D000004

APPENDIX I
MWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

\$ ADA E000001
\$ ADA E000002
\$ ADA E000003
\$ ADA E000004
\$ ADA E000005
\$ ADA F000001
\$ ADA F000002
\$ ADA G000001
\$ ADA G000002
\$ ADA G000003
\$ ADA G000004
\$ ADA G000005
\$ ADA G000006
\$ ADA G000007
\$ ADA H000001
\$ ADA H000002
\$ ADA H000003
\$ ADA H000004
\$ ADA H000005
\$ ADA H000006
\$ ADA H000007
\$ ADA H000008
\$ ADA H000009
\$ ADA L000001
\$ ADA L000002
\$ ADA L000003
\$ ADA L000004
\$ ADA L000005
\$ ADA P000001
\$ ADA P000002
\$ ADA P000003
\$ ADA P000004
\$ ADA P000005
\$ ADA P000006
\$ ADA P000007
\$ ADA P000010
\$ ADA P000011
\$ ADA P000012
\$ ADA P000013
\$ ADA T000001
\$ ADA T000002
\$ ADA T000003
\$ ADA T000004
\$ ADA T000005
\$ ADA T000006
\$ ADA T000007 !execution possibly machine dependent

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$ ADA T000008
$ ADA Y000001
$ ADA Y000002
$ ADA Y000003
$ ADA A000100
$ ACS LINK A000100
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK:= ""
$LINK-
/NOMAP-
/EXE=[A000100-
SYSS$INPUT:/OPTIONS
$!$DIA9:[DAVIDSON.PIWG]A000100.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000100.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000100.COM;1
$EXIT
$ RUN A000100 !all feature tests in one procedure
0.0201 = clock resolution used for iteration stability

```

Test Name: A000090

Clock resolution measurement running

Test Description: Determine clock resolution using second differences of values returned by the function CPU_Time_Clock.

Number of sample values is	12000
Clock Resolution	= 0.009948730468750 seconds.
Clock Resolution (average)	= 0.009948730468750 seconds.
Clock Resolution (variance)	= 0.0000000000000000 seconds.

Capture a few sizes for later analysis

```

1 Boolean's size
8 Boolean_object's size
8 Character's size
8 Character_object's size
31 Positive's size
32 Positive_object's size
32 Integer's size
32 Float's size
32 Duration's size
0.0001 Duration's small
0.0099 System.Tick

```

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: A000091
CPU Time: 156.25 MICROSECONDS
Wall/CPU: 1.12 ratio.
Test Description: Reinhold P. Weicker's DHRYSTONE composite benchmark.

Class Name: Composite
plus or minus 7.813
Iteration Count: 25600

Test Name: A000093
CPU Time: 175.00 MILLISECONDS
Wall/CPU: 1.07 ratio.
Test Description: Ada version of the Whetstone Benchmark Program. Built in 'standard' math routines.

Class Name: Composite
plus or minus 8.750
Iteration Count: 40

Average Whetstone rating: 5714 KWIPS

Test Name: A000094

Class: Composite

Perm 0.45
Towers 0.72
Queens 0.23
Intmm 0.19
Mm 0.18
Puzzle 1.09
Quick 0.18
Bubble 0.25
Tree 0.36
FFT 0.30
Ack 16.42

Test Description: Henessy benchmarks.

Test Name: A000094A
CPU Time: 0.44 SECONDS
Wall/CPU: 1.11 ratio.

Class Name: Composite
plus or minus 0.022
Iteration Count: 10

Test Description: Hennesy Benchmark, Perm highly recursive 43300 uses of procedure permute.

Test Name: A000094B
CPU Time: 0.74 SECONDS
Wall/CPU: 1.07 ratio.

Class Name: Composite
plus or minus 0.037
Iteration Count: 10

Test Description: Hennesy Benchmark, TOWERS highly recursive.

Test Name: A000094C
CPU Time: 0.24 SECONDS
Wall/CPU: 1.03 ratio.

Class Name: Composite
plus or minus 0.012
Iteration Count: 20

Test Description: Hennesy Benchmark, Queens highly recursive.

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: A000094D	Class Name: Composite
CPU Time: 0.18 SECONDS	plus or minus 0.009
Wall/CPU: 1.09 ratio.	Iteration Count: 40
Test Description: Hennessy Benchmark, Intmm integer matrix multiply performed by an instantiated generic procedure.	
Test Name: A000094E	Class Name: Composite
CPU Time: 0.19 SECONDS	plus or minus 0.010
Wall/CPU: 1.04 ratio.	Iteration Count: 40
Test Description: Hennessy Benchmark, Mm real matrix multiply performed by an instantiated generic procedure.	
Test Name: A000094F	Class Name: Composite
CPU Time: 1.12 SECONDS	plus or minus 0.056
Wall/CPU: 1.05 ratio.	Iteration Count: 10
Test Description: Hennessy Benchmark, Puzzle highly recursive.	
Test Name: A000094G	Class Name: Composite
CPU Time: 0.18 SECONDS	plus or minus 0.009
Wall/CPU: 1.08 ratio.	Iteration Count: 40
Test Description: Hennessy Benchmark, quick sort 5000 element random integer array using quick sort.	
Test Name: A000094H	Class Name: Composite
CPU Time: 0.25 SECONDS	plus or minus 0.012
Wall/CPU: 1.02 ratio.	Iteration Count: 20
Test Description: Hennessy Benchmark, Bubble sort 5000 random integer array using bubble sort.	
Test Name: A000094I	Class Name: Composite
CPU Time: 0.37 SECONDS	plus or minus 0.019
Wall/CPU: 1.07 ratio.	Iteration Count: 20
Test Description: Hennessy Benchmark, Tree insert 5000 random elements into a tree that is sorted.	
Test Name: A000094J	Class Name: Composite
CPU Time: 0.27 SECONDS	plus or minus 0.013
Wall/CPU: 1.03 ratio.	Iteration Count: 16
Test Description: Hennessy Benchmark, FFT perform 20 256 point complex FFT's floating point intensive.	
Test Name: A000094K	Class Name: Composite
CPU Time: 16.42 SECONDS	plus or minus 0.821
Wall/CPU: 1.05 ratio.	Iteration Count: 2
Test Description: Hennessy Benchmark, Ack highly recursive 10 executions of Ackerman (3,6).	

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: B000001A **Class Name:** Application
CPU Time: 1017.00 MICROSECONDS plus or minus 50.850
Wall/CPU: 1.05 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix fixed point, delta
2.0**(-15) all checks on.

Test Name: B000001B **Class Name:** Application
CPU Time: 546.00 MICROSECONDS plus or minus 27.300
Wall/CPU: 1.01 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix fixed point, delta
2.0**(-15) checks suppressed.

Test Name: B000002A **Class Name:** Application
CPU Time: 462.00 MICROSECONDS plus or minus 23.100
Wall/CPU: 1.13 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 6 range -
1.0e+9..1.0e+9 all checks on.

Test Name: B000002B **Class Name:** Application
CPU Time: 311.00 MICROSECONDS plus or minus 15.550
Wall/CPU: 1.09 ratio. Iteration Count: 20000
Test Description: Tracking mathematical application using covariance matrix digits 6 range -
1.0e+9..1.0e+9 checks suppressed.

Test Name: B000003A **Class Name:** Application
CPU Time: 781.00 MICROSECONDS plus or minus 39.050
Wall/CPU: 1.04 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 9 range -
1.0e+9..1.0e+9 all checks on.

Test Name: B000003B **Class Name:** Application
CPU Time: 514.00 MICROSECONDS plus or minus 25.700
Wall/CPU: 1.10 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 9 range -
1.0e+9..1.0e+9 checks suppressed.

Test Name: B000004A **Class Name:** Application
CPU Time: 484.00 MICROSECONDS plus or minus 24.200
Wall/CPU: 1.09 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 6 and
integer all checks on.

APPENDIX I

Test Name: B000004B	Class Name: Application
CPU Time: 327.00 MICROSECONDS	plus or minus 16.350
Wall/CPU: 1.03 ratio.	Iteration Count: 20000
Test Description: Tracking mathematical application using covariance matrix digits 6 and integer checks suppressed.	

Test Name: B000010	Class Name: Composite
CPU Time: 939.99 MILLISECONDS	plus or minus 46.999
Wall/CPU: 1.12 ratio.	Iteration Count: 5
Test Description: NASA Orbit determination heavy floating point and trig initializing and running.	

4.75433302E+03 4.65319583E+03 2.26151483E+02
-5.29274842E+00 5.22249948E+00 2.25443633E+00

Test Name: B000011	Class Name: Composite
CPU Time: 85.37 MILLISECONDS	plus or minus 4.269
Wall/CPU: 1.03 ratio.	Iteration Count: 80
Test Description: JIAWG Kalman benchmark. Just put in PIWG measurement harness.	

Test Name: B000013	Class Name: Composite
CPU Time: 7.42 MILLISECONDS	plus or minus 0.371
Wall/CPU: 1.04 ratio.	Iteration Count: 800
Test Description: TRACKER CENTROID Algorithm all integer calculations searching a 60 x 60 array.	

Test Name: C000001	Class Name: Tasking
CPU Time: 1890.66 MICROSECONDS	plus or minus 94.533
Wall/CPU: 1.06 ratio.	Iteration Count: 3200
Test Description: Task create and terminate measurement with one task, no entries, when task is in a procedure using a task type in a package, no select statement, no loop.	

Test Name: C000002	Class Name: Tasking
CPU Time: 1893.75 MICROSECONDS	Plus or minus: 94.687
Wall/CPU: 1.07 ratio.	Iteration Count: 3200
Test Description: Task create and terminate time measurement, with one task, no entries when task is in a procedure, task defined and used in procedure, no select statement, no loop.	

Test Name: C000003	Class Name: Tasking
CPU Time: 1893.75 MICROSECONDS	plus or minus 94.687
Wall/CPU: 1.07 ratio.	Iteration Count: 3200
Test Description: Task create and terminate time measurement. Task is in declare block of main procedure one task, no entries, task is in the loop.	

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: D000001 Class Name: Allocation
 CPU Time: 2.91 MICROSECONDS plus or minus 0.197
 Wall/CPU: 1.03 ratio. Iteration Count: 409600
 Test Description: Dynamic array allocation, use and deallocation time measurement.
 Dynamic array elaboration, 1000 integers in a procedure get space and free it
 in the procedure on each call.

Test Name: D000002 Class Name: Allocation
 CPU Time: 535.94 MICROSECONDS plus or minus 26.797
 Wall/CPU: 1.04 ratio. Iteration Count: 12800
 Test Description: Dynamic array elaboration and initialization time measurement allocation,
 initialization, use and deallocation 1000 integers initialized by others=>1.

Test Name: D000003 Class Name: Allocation
 CPU Time: 3.00 MICROSECONDS plus or minus 0.197
 Wall/CPU: 1.09 ratio. Iteration Count: 409600
 Test Description: Dynamic record allocation and deallocation time measurement elaborating,
 allocating and deallocating record containing a dynamic array of 1000
 integers.

Test Name: D000004 Class Name: Allocation
 CPU Time: 542.96 MICROSECONDS plus or minus 27.148
 Wall/CPU: 1.02 ratio. Iteration Count: 12800
 Test Description: Dynamic record allocation and deallocation time measurement elaborating,
 initializing by (DYNAMIC_SIZE, (others=>1)) record containing a dynamic
 array of 1000 integers

Test Name: E000001 Class Name: Exception
 CPU Time: 86.91 MICROSECONDS plus or minus 4.346
 Wall/CPU: 1.03 ratio. Iteration Count: 51200
 Test Description: Time to raise and handle an exception. Exception defined locally and handled
 locally.

Test Name: E000002 Class Name: Exception
 CPU Time: 145.70 MICROSECONDS plus or minus 7.285
 Wall/CPU: 1.01 ratio. Iteration Count: 25600
 Test Description: Exception raise and handle timing measurement when exception is in a
 procedure in a package.

Test Name: E000003 Class Name: Exception
 CPU Time: 98.05 MICROSECONDS plus or minus 4.902
 Wall/CPU: 1.03 ratio. Iteration Count: 51200
 Test Description: Exception raise and handle timing measurement when exception is raised
 nested 3 deep in procedure calls.

APPENDIX I

Test Name: E000004	Class Name: Exception
CPU Time: 97.46 MICROSECONDS	plus or minus 4.873
Wall/CPU: 1.02 ratio.	Iteration Count: 51200
Test Description: Exception raise and handle timing measurement when exception is nested 4 deep in procedures.	

Test Name: E000005	Class Name: Exception
CPU Time: 982.81 MICROSECONDS	plus or minus 49.140
Wall/CPU: 1.03 ratio.	Iteration Count: 6400
Test Description: Exception raise and handle timing measurement when exception is in a rendezvous. Both the task and the caller must handle the exception.	

Test Name: F000001	Class Name: Style
CPU Time: 0.55 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.03 ratio.	Iteration Count: 819200
Test Description: Time to set a boolean flag using a logical equation a local and a global integer are compared compare this test with F000002.	

Test Name: F000002	Class Name: Style
CPU Time: 0.43 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.04 ratio.	Iteration Count: 819200
Test Description: Time to set a boolean flag using an 'if' test a local and a global integer are compared compare this test with F000001.	

Test Name: G000001	Class Name: Input/Output
CPU Time: 114.01 MICROSECONDS	plus or minus 5.701
Wall/CPU: 1.02 ratio.	Iteration Count: 40960
Test Description: TEXT_IO.GET_LINE reading 20 characters, time measured. A scratch file is written, then read and reset	

Test Name: G000002	Class Name: Input/Output
CPU Time: 765.63 MICROSECONDS	plus or minus 38.282
Wall/CPU: 1.03 ratio.	Iteration Count: 5120
Test Description: TEXT_IO.GET called 20 times per line, time measured a scratch file is written, then read and reset. Compare to G000001 for about same number of characters.	

Test Name: G000003	Class Name: Input/Output
CPU Time: 818.37 MICROSECONDS	plus or minus 300.902
Wall/CPU: 19.12 ratio.	Iteration Count: 5120
Test Description: TEXT_IO.PUT_LINE for 20 characters, timing measurement a scratch file is opened, written and reset.	

APPENDIX I

Test Name: G000004	Class Name: Input/Output
CPU Time: 1343.76 MICROSECONDS	plus or minus 182.355
Wall/CPU: 11.59 ratio.	Iteration Count: 5120
Test Description: TEXT_IO.PUT 20 times with one character, time measurement a scratch file is written, reset and rewritten compare, approximately, to G000003.	

Test Name: G000005	Class Name: Input/Output
CPU Time: 70.80 MICROSECONDS	plus or minus 3.540
Wall/CPU: 1.03 ratio.	Iteration Count: 40960
Test Description: TEXT_IO.GET an integer from a local string, timing measurement use TEXT_IO.PUT to convert 1..100 to a string then use TEXT_IO.GET to get the number back.	

Test Name: G000006	Class Name: Input/Output
CPU Time: 162.11 MICROSECONDS	plus or minus 8.105
Wall/CPU: 1.06 ratio.	Iteration Count: 20480
Test Description: TEXT_IO.GET getting a floating point fraction from a local string. Timing measurement on 0.001 to 0.01 range of numbers compare, approximately, to G000005 for integer vs float.	

Test Name: G000007	Class Name: Input/Output
CPU Time: 18968.96 MICROSECONDS	plus or minus 861.479
Wall/CPU: 3.42 ratio.	Iteration Count: 320
Test Description: Open and close an existing file, time measurement. A scratch file is created and closed. The scratch file is opened IN_FILE and closed in a loop.	

Test Name: H000001	Class Name: Chapter 13
CPU Time: 1.79 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.04 ratio.	Iteration Count: 819200
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are PACKED with the pragma 'PACK.' For this test the operations are performed on the entire arrays.	

Test Name: H000002	Class Name: Chapter 13
CPU Time: 41.70 MICROSECONDS	plus or minus 2.085
Wall/CPU: 1.07 ratio.	Iteration Count: 102400
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are NOT PACKED with the pragma 'PACK.' For this test the operations are performed on the entire array.	

Test Name: H000003	Class Name: Chapter 13
CPU Time: 130.47 MICROSECONDS	plus or minus 6.523
Wall/CPU: 1.04 ratio.	Iteration Count: 51200
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are PACKED with the pragma 'PACK.' For this test the operations are performed on components in a loop.	

APPENDIX I

Test Name: H000004	Class Name: Chapter 13
CPU Time: 44.63 MICROSECONDS	plus or minus 2.231
Wall/CPU: 1.05 ratio.	Iteration Count: 102400
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are NOT PACKED with the pragma 'PACK.' For this test the operations are performed on components in a loop.	

Test Name: H000005	Class Name: Chapter 13
CPU Time: 0.22 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.05 ratio.	Iteration Count: 819200
Test Description: The time for UNCHECKED_CONVERSION to move one INTEGER object to another INTEGER object. This may be zero with good optimization.	

Test Name: H000006	Class Name: Chapter 13
CPU Time: 7.42 MICROSECONDS	plus or minus 0.371
Wall/CPU: 1.04 ratio.	Iteration Count: 409600
Test Description: The time for UNCHECKED_CONVERSION to move 10 floating array objects to a 10 component floating record.	

Test Name: H000007	Class Name: Chapter 13
CPU Time: 11.01 MICROSECONDS	plus or minus 0.551
Wall/CPU: 1.03 ratio.	Iteration Count: 409600
Test Description: The time to store and extract bit fields that are defined by representation clauses using Boolean and Integer record components. 12 accesses, 5 stores, 1 record copy.	

Test Name: H000008	Class Name: Chapter 13
CPU Time: 4.08 MICROSECONDS	plus or minus 0.204
Wall/CPU: 1.05 ratio.	Iteration Count: 409600
Test Description: The time to store and extract bit fields that are defined by nested representation clauses using packed arrays of Boolean and Integer record components.	

Test Name: H000009	Class Name: Chapter 13
CPU Time: 13.06 MICROSECONDS	plus or minus 0.653
Wall/CPU: 1.04 ratio.	Iteration Count: 409600
Test Description: The time to perform a change of representation. If the result is near zero, feature probably not implemented	

Test Name: L000001	Class Name: Iteration
CPU Time: 0.25 MICROSECONDS	plus or minus 0.063
Wall/CPU: 1.04 ratio.	Iteration Count: 1280000
Test Description: Simple "for" loop time for I in 1 .. 100 loop time reported is for once through loop.	

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: L000002	Class Name: Iteration
CPU Time: 0.16 MICROSECONDS	plus or minus 0.063
Wall/CPU: 1.06 ratio.	Iteration Count: 1280000
Test Description: Simple "while" loop time while I <= 100 loop time reported is for once through loop.	
Test Name: L000003	Class Name: Iteration
CPU Time: 0.19 MICROSECONDS	plus or minus 0.063
Wall/CPU: 1.07 ratio.	Iteration Count: 1280000
Test Description: Simple "exit" loop time; loop I:=I+1; exit when I>100; end loop; time reported is for once through loop.	
Test Name: L000004	Class Name: Iteration
CPU Time: 0.21 MICROSECONDS	plus or minus 0.079
Wall/CPU: 1.18 ratio.	Iteration Count: 1024000
Test Description: Measures Compiler's choice to UNWRAP a small loop of five (5) iterations when given a PRAGMA OPTIMIZE (TIME). An execution time less than .05 microseconds indicates the unwrap occurred.	
Test Name: L000005	Class Name: Iteration
CPU Time: 0.19 MICROSECONDS	plus or minus 0.079
Wall/CPU: 1.06 ratio.	Iteration Count: 1024000
Test Description: Measures Compiler's choice to UNWRAP a small loop of five (5) iterations when given a PRAGMA OPTIMIZE (Space). An execution speed < .05 microseconds indicates the unwrap occurred.	
Test Name: P000001	Class Name: Procedure
CPU Time: 0.06 MICROSECONDS	plus or minus 0.079
Wall/CPU: 1.05 ratio.	Iteration Count: 1024000
Test Description: Procedure call and return time (may be zero if automatic inlining) procedure is local no parameters.	
Test Name: P000002	Class Name: Procedure
CPU Time: 7.01 MICROSECONDS	plus or minus 0.351
Wall/CPU: 1.06 ratio.	Iteration Count: 512000
Test Description: Procedure call and return time. Procedure is local, no parameters when procedure is not inlinable.	
Test Name: P000003	Class Name: Procedure
CPU Time: 7.07 MICROSECONDS	plus or minus 0.353
Wall/CPU: 1.08 ratio.	Iteration Count: 512000
Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. Compare to P000002.	

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: P000013	Class Name: Procedure
CPU Time: 13.87 MICROSECONDS	plus or minus 0.693
Wall/CPU: 1.08 ratio.	Iteration Count: 256000
Test Description: Procedure call and return time measurement 20 composite 'in' parameters the composite type is a three component record	

Test Name: T000001	Class Name: Tasking
CPU Time: 175.78 MICROSECONDS	plus or minus 8.789
Wall/CPU: 1.08 ratio.	Iteration Count: 25600
Test Description: Minimum rendezvous, entry call and return time 1 task 1 entry , task inside procedure no select.	

Test Name: T000002	Class Name: Tasking
CPU Time: 171.48 MICROSECONDS	plus or minus 8.574
Wall/CPU: 1.04 ratio.	Iteration Count: 25600
Test Description: Task entry call and return time measured. One task active, one entry in task, task in a package, no select statement.	

Test Name: T000003	Class Name: Tasking
CPU Time: 183.59 MICROSECONDS	plus or minus 9.180
Wall/CPU: 1.10 ratio.	Iteration Count: 25600
Test Description: Task entry call and return time measured. Two tasks active, one entry per task, tasks in a package, no select statement.	

Test Name: T000004	Class Name: Tasking
CPU Time: 205.86 MICROSECONDS	plus or minus 10.293
Wall/CPU: 1.05 ratio.	Iteration Count: 25600
Test Description: Task entry call and return time measured. One tasks active, two entries, tasks in a package using select statement.	

Test Name: T000005	Class Name: Tasking
CPU Time: 230.00 MICROSECONDS	plus or minus 11.500
Wall/CPU: 1.11 ratio.	Iteration Count: 32000
Test Description: Task entry call and return time measured. 10 tasks active, one entry per task, tasks in a package, no select statement.	

Test Name: T000006	Class Name: Tasking
CPU Time: 321.25 MICROSECONDS	plus or minus 16.062
Wall/CPU: 1.06 ratio.	Iteration Count: 16000
Test Description: Task entry call and return time measurement. One task with ten entries, task in a package one select statement, compare to T000005.	

APPENDIX I

Test Name: T000008	Class Name: Tasking
CPU Time: 375.01 MICROSECONDS	<i>plus or minus</i> 18.750
Wall/CPU: 1.09 ratio.	Iteration Count: 12800
Test Description: Measure the average time to pass an integer from a producer task through a buffer task to a consumer task.	

Test Name: Y000001 Measure actual delay vs commanded delay

Commanded	Actual	CPU	Iterations
0.0010	0.0100	0.0000	1024
0.0020	0.0101	0.0000	512
0.0039	0.0107	0.0000	256
0.0078	0.0099	0.0000	128
0.0156	0.0200	0.0000	64
0.0313	0.0400	0.0000	32
0.0625	0.0699	0.0000	16
0.1250	0.1299	0.0000	8
0.2500	0.2500	0.0000	4
0.5000	0.5000	0.0000	2
1.0000	1.0000	0.0000	2
2.0000	2.0000	0.0000	2

```

$!          !run one feature at a time if results are better
$!          !mail in printout
$ ADA A000101 !about half the features
$ ADA A000102 !the other half
$ ACS LINK A000101
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000101-
        SYSS$INPUT:/OPTIONS
        $!$DIA9:[DAVIDSON.PIWG]A000101.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000101.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000101.COM;1
$EXIT
$ ACS LINK A000102
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-

```

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

```
/EXE=[]A000102-
  SYSS$INPUT:/OPTIONS
  $1$DIA9:[DAVIDSON.PIWG]A000102.OBJ;1
$STATUS = $STATUS
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000102.OBJ;1
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000102.COM;1
$EXIT
$ ACS LINK A000051
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $1$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000051-
  SYSS$INPUT:/OPTIONS
  $1$DIA9:[DAVIDSON.PIWG]A000051.OBJ;1
$STATUS = $STATUS
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000051.OBJ;1
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000051.COM;1
$EXIT
$ ACS LINK A000052
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $1$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000052-
  SYSS$INPUT:/OPTIONS
  $1$DIA9:[DAVIDSON.PIWG]A000052.OBJ;1
$STATUS = $STATUS
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000052.OBJ;1
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000052.COM;1
$EXIT
$ ACS LINK A000053
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $1$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000053-
  SYSS$INPUT:/OPTIONS
  $1$DIA9:[DAVIDSON.PIWG]A000053.OBJ;1
$STATUS = $STATUS
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000053.OBJ;1
$DELETE $1$DIA9:[DAVIDSON.PIWG]A000053.COM;1
$EXIT
```

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$ ACS LINK A000054
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000054-
    SYSS$INPUT:/OPTIONS
    $!$DIA9:[DAVIDSON.PIWG]A000054.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000054.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000054.COM;1
$EXIT
$ ACS LINK A000055
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000055-
    SYSS$INPUT:/OPTIONS
    $!$DIA9:[DAVIDSON.PIWG]A000055.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000055.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000055.COM;1
$EXIT
$ ACS LINK A000095
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000095-
    SYSS$INPUT:/OPTIONS
    $!$DIA9:[DAVIDSON.PIWG]A000095.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000095.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000095.COM;1
$EXIT
$ ACS LINK Y000002
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]Y000002-

```

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

SYSS\$INPUT:/OPTIONS

```
$1$DIA9:[DAVIDSON.PIWG]Y000002.OBJ;1
$STATUS = $STATUS
$DELETE $1$DIA9:[DAVIDSON.PIWG]Y000002.OBJ;1
$DELETE $1$DIA9:[DAVIDSON.PIWG]Y000002.COM;1
$EXIT
$ ACS LINK Y000003
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $1$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[Y000003-
```

SYSS\$INPUT:/OPTIONS

```
$1$DIA9:[DAVIDSON.PIWG]Y000003.OBJ;1
$STATUS = $STATUS
$DELETE $1$DIA9:[DAVIDSON.PIWG]Y000003.OBJ;1
$DELETE $1$DIA9:[DAVIDSON.PIWG]Y000003.COM;1
$EXIT
$ RUN A000051
    CPU time now= 1689.3900    WALL time now= 65495.6900 seconds.
$ RUN A000052
$ RUN A000053
$ RUN A000054
$ RUN A000101
```

0.0201 = clock resolution used for iteration stability

Test Name: A000090

Clock resolution measurement running

Test Description: Determine clock resolution using second differences of values returned by the function CPU_Time_Clock.

```
Number of sample values is    12000
Clock Resolution              =    0.009948730468750 seconds.
Clock Resolution (average) =    0.009948730468750 seconds.
Clock Resolution (variance) =    0.000000000000000 seconds.
```

Capture a few sizes for later analysis

```
1 Boolean's size
8 Boolean_object's size
8 Character's size
8 Character_object's size
31 Positive's size
32 Positive_object's size
32 Integer's size
32 Float's size
```

APPENDIX I
MTWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

32 Duration'size

0.0001 Duration'small

0.0099 System.Tick

Test Name: A000091 Class Name: Composite
CPU Time: 171.09 MICROSECONDS plus or minus 8.555
Wall/CPU: 1.10 ratio. Iteration Count: 25600
Test Description: Reinhold P. Weicker's DHRYSTONE composite benchmark.

Test Name: A000092 Class Name: Composite
CPU Time: 142.50 MILLISECONDS plus or minus 7.125
Wall/CPU: 1.07 ratio. Iteration Count: 40
Test Description: Ada version of the Whetstone Benchmark Program. Manufacturers math routines

Average Whetstone rating: 7018 KWIPS

Test Name: A000093 Class Name: Composite
CPU Time: 178.25 MILLISECONDS plus or minus 8.912
Wall/CPU: 1.04 ratio. Iteration Count: 40
Test Description: Ada version of the Whetstone Benchmark Program. Built in 'standard' math routines.

Average Whetstone rating: 5610 KWIPS

Test Name: A000094 Class: Composite
Perm 0.48
Towers 0.72
Queens 0.24
Intmm 0.19
Mm 0.22
Puzzle 1.11
Quick 0.18
Bubble 0.24
Tree 0.41
FFT 0.30
Ack 16.40

Test Description: Henessy benchmarks.

Test Name: A000094A Class Name: Composite
CPU Time: 0.43 SECONDS plus or minus 0.022
Wall/CPU: 1.11 ratio. Iteration Count: 10
Test Description: Hennessy Benchmark, Perm highly recursive 43300 uses of procedure permute.

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: A000094B	Class Name: Composite
CPU Time: 0.73 SECONDS	plus or minus 0.037
Wall/CPU: 1.09 ratio.	Iteration Count: 10
Test Description: Hennessy Benchmark, TOWERS highly recursive.	
Test Name: A000094C	Class Name: Composite
CPU Time: 0.24 SECONDS	plus or minus 0.012
Wall/CPU: 1.06 ratio.	Iteration Count: 20
Test Description: Hennessy Benchmark, Queens highly recursive.	
Test Name: A000094D	Class Name: Composite
CPU Time: 0.17 SECONDS	plus or minus 0.009
Wall/CPU: 1.03 ratio.	Iteration Count: 40
Test Description: Hennessy Benchmark, Intrmm integer matrix multiply performed by an instantiated generic procedure.	
Test Name: A000094E	Class Name: Composite
CPU Time: 0.19 SECONDS	plus or minus 0.010
Wall/CPU: 1.05 ratio.	Iteration Count: 40
Test Description: Hennessy Benchmark, Mm real matrix multiply performed by an instantiated generic procedure.	
Test Name: A000094F	Class Name: Composite
CPU Time: 1.10 SECONDS	plus or minus 0.055
Wall/CPU: 1.03 ratio.	Iteration Count: 10
Test Description: Hennessy Benchmark, Puzzle highly recursive.	
Test Name: A000094G	Class Name: Composite
CPU Time: 0.18 SECONDS	plus or minus 0.009
Wall/CPU: 1.09 ratio.	Iteration Count: 40
Test Description: Hennessy Benchmark, quick sort 5000 element random integer array using quick sort.	
Test Name: A000094H	Class Name: Composite
CPU Time: 0.25 SECONDS	plus or minus 0.012
Wall/CPU: 1.12 ratio.	Iteration Count: 20
Test Description: Hennessy Benchmark, Bubble sort 5000 random integer array using bubble sort.	
Test Name: A000094I	Class Name: Composite
CPU Time: 0.38 SECONDS	plus or minus 0.019
Wall/CPU: 1.09 ratio.	Iteration Count: 20
Test Description: Hennessy Benchmark, Tree insert 5000 random elements into a tree that is sorted.	

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: A000094J Class Name: Composite
CPU Time: 0.28 SECONDS plus or minus 0.014
Wall/CPU: 1.09 ratio. Iteration Count: 16
Test Description: Hennessy Benchmark, FFT perform 20 256 point complex FFT's floating point intensive.

Test Name: A000094K Class Name: Composite
CPU Time: 16.57 SECONDS plus or minus 0.058
Wall/CPU: 1.43 ratio. Iteration Count: 2
Test Description: Hennessy Benchmark, Ack highly recursive 10 executions of Ackerman (3,6)

Test Name: B000001A Class Name: Application
CPU Time: 829.00 MICROSECONDS plus or minus 23.100
Wall/CPU: 2.87 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix fixed point, delta 2.0**(-15) all checks on.

Test Name: B000001B Class Name: Application
CPU Time: 550.99 MICROSECONDS plus or minus 12.195
Wall/CPU: 1.51 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix fixed point, delta 2.0**(-15) checks suppressed.

Test Name: B000002A Class Name: Application
CPU Time: 441.00 MICROSECONDS plus or minus 11.491
Wall/CPU: 1.43 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 6 range - 1.0e+9..1.0e+9 all checks on.

Test Name: B000002B Class Name: Application
CPU Time: 313.00 MICROSECONDS plus or minus 15.650
Wall/CPU: 1.04 ratio. Iteration Count: 20000
Test Description: Tracking mathematical application using covariance matrix digits 6 range - 1.0e+9..1.0e+9 checks suppressed.

Test Name: B000003A Class Name: Application
CPU Time: 880.00 MICROSECONDS plus or minus 10.098
Wall/CPU: 1.25 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 9 range - 1.0e+9..1.0e+9 all checks on.

Test Name: B000003B Class Name: Application
CPU Time: 509.00 MICROSECONDS plus or minus 25.450
Wall/CPU: 1.04 ratio. Iteration Count: 10000
Test Description: Tracking mathematical application using covariance matrix digits 9 range - 1.0e+9..1.0e+9 checks suppressed.

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: B000004A Class Name: Application
 CPU Time: 478.00 MICROSECONDS plus or minus 23.900
 Wall/CPU: 1.01 ratio. Iteration Count: 10000
 Test Description: Tracking mathematical application using covariance matrix digits 6 and integer all checks on.

Test Name: B000004B Class Name: Application
 CPU Time: 322.50 MICROSECONDS plus or minus 16.125
 Wall/CPU: 1.03 ratio. Iteration Count: 20000
 Test Description: Tracking mathematical application using covariance matrix digits 6 and integer checks suppressed.

Test Name: B000010 Class Name: Composite
 CPU Time: 940.00 MILLISECONDS plus or minus 47.000
 Wall/CPU: 1.03 ratio. Iteration Count: 5
 Test Description: NASA Orbit determination heavy floating point and trig initializing and running.

4.75433302E+03 4.65319583E+03 2.26151483E+02
 -5.29274842E+00 5.22249948E+00 2.25443633E+00

Test Name: B000011 Class Name: Composite
 CPU Time: 87.25 MILLISECONDS plus or minus 2.990
 Wall/CPU: 2.97 ratio. Iteration Count: 80
 Test Description: JIAWG Kalman benchmark. Just put in PIWG measurement harness.

Test Name: B000013 Class Name: Composite
 CPU Time: 7.47 MILLISECONDS plus or minus 0.374
 Wall/CPU: 1.01 ratio. Iteration Count: 800
 Test Description: TRACKER CENTROID Algorithm. All integer calculations searching a 60 x 60 array.

Test Name: C000001 Class Name: Tasking
 CPU Time: 1943.74 MICROSECONDS plus or minus 31.937
 Wall/CPU: 1.27 ratio. Iteration Count: 3200
 Test Description: Task create and terminate measurement with one task, no entries, when task is in a procedure using a task type in a package, no select statement, no loop.

Test Name: C000002 Class Name: Tasking
 CPU Time: 1903.15 MICROSECONDS plus or minus 30.451
 Wall/CPU: 1.21 ratio. Iteration Count: 3200
 Test Description: Task create and terminate time measurement, with one task, no entries when task is in a procedure, task defined and used in procedure, no select statement, no loop.

APPENDIX I

Test Name: C000003

Class Name: Tasking

CPU Time: 1890.62 MICROSECONDS

plus or minus 94.531

Wall/CPU: 1.06 ratio.

Iteration Count: 3200

Test Description: Task create and terminate time measurement. Task is in declare block of main procedure one task, no entries, task is in the loop.

Test Name: D000001

Class Name: Allocation

CPU Time: 2.56 MICROSECONDS

plus or minus 0.197

Wall/CPU: 5.06 ratio.

Iteration Count: 409600

Test Description: Dynamic array allocation, use and deallocation time measurement.
Dynamic array elaboration, 1000 integers in a procedure get space and free it in the procedure on each call.

Test Name: D000002

Class Name: Allocation

CPU Time: 550.00 MICROSECONDS

plus or minus 9.169

Wall/CPU: 1.46 ratio.

Iteration Count: 12800

Test Description: Dynamic array elaboration and initialization time measurement
allocation, initialization, use and deallocation 1000 integers initialized by
others=>1.

Test Name: D000003

Class Name: Allocation

CPU Time: 2.78 MICROSECONDS

plus or minus 0.197

Wall/CPU: 1.07 ratio.

Iteration Count: 409600

Test Description: Dynamic record allocation and deallocation time measurement elaborating, allocating and deallocating record containing a dynamic array of 1000 integers.

Test Name: D000004

Class Name: Allocation

CPU Time: 548.43 MICROSECONDS

plus or minus 27.422

Wall/CPU: 1.19 ratio.

Iteration Count: 12800

Test Description: Dynamic record allocation and deallocation time measurement elaborating, initializing by (DYNAMIC_SIZE,(others=>1)) record containing a dynamic array of 1000 integers.

\$ RUN A000102

0.0201 = clock resolution used for iteration stability

Test Name: E000001

Class Name: Exception

CPU Time: 88.48 MICROSECONDS

plus or minus 4.424

Wall/CPU: 1.03 ratio.

Iteration Count: 51200

Test Description: Time to raise and handle an exception. Exception defined locally and handled locally.

APPENDIX I

Test Name: E000002	Class Name: Exception
CPU Time: 146.49 MICROSECONDS	plus or minus 7.324
Wall/CPU: 1.07 ratio.	Iteration Count: 25600
Test Description: Exception raise and handle timing measurement when exception is in a procedure in a package.	

Test Name: E000003	Class Name: Exception
CPU Time: 98.44 MICROSECONDS	plus or minus 4.922
Wall/CPU: 1.05 ratio.	Iteration Count: 51200
Test Description: Exception raise and handle timing measurement when exception is raised nested 3 deep in procedure calls.	

Test Name: E000004	Class Name: Exception
CPU Time: 98.63 MICROSECONDS	plus or minus 4.932
Wall/CPU: 1.06 ratio.	Iteration Count: 51200
Test Description: Exception raise and handle timing measurement when exception is nested 4 deep in procedures.	

Test Name: E000005	Class Name: Exception
CPU Time: 995.30 MICROSECONDS	plus or minus 49.765
Wall/CPU: 1.09 ratio.	Iteration Count: 6400
Test Description: Exception raise and handle timing measurement when exception is in a rendezvous. Both the task and the caller must handle the exception.	

Test Name: F000001	Class Name: Style
CPU Time: 0.54 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.06 ratio.	Iteration Count: 819200
Test Description: Time to set a boolean flag using a logical equation a local and a global integer are compared compare this test with F000002.	

Test Name: F000002	Class Name: Style
CPU Time: 0.46 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.08 ratio.	Iteration Count: 819200
Test Description: Time to set a boolean flag using an 'if' test a local and a global integer are compared. Compare this test with F000001.	

Test Name: G000001	Class Name: Input/Output
CPU Time: 110.11 MICROSECONDS	plus or minus 2.685
Wall/CPU: 1.37 ratio.	Iteration Count: 40960
Test Description: TEXT_IO.GET_LINE reading 20 characters, time measured. A scratch file is written, then read and reset.	

APPENDIX I

Test Name: G000002	Class Name: Input/Output
CPU Time: 769.54 MICROSECONDS	plus or minus 38.477
Wall/CPU: 1.07 ratio.	Iteration Count: 5120
Test Description: TEXT_IO.GET called 20 times per line, time measured a scratch file is written, then read and reset. Compare to G000001 for about same number of characters.	

Test Name: G000003	Class Name: Input/Output
CPU Time: 832.04 MICROSECONDS	plus or minus 166.720
Wall/CPU: 21.19 ratio.	Iteration Count: 10240
Test Description: TEXT_IO.PUT_LINE for 20 characters, timing measurment a scratch file is opened, written and reset.	

Test Name: G000004	Class Name: Input/Output
CPU Time: 1367.20 MICROSECONDS	plus or minus 167.903
Wall/CPU: 10.67 ratio.	Iteration Count: 5120
Test Description: TEXT_IO.PUT 20 times with one character, time measurement a scratch file is written, reset and rewritten. Compare, approximately, to G000003.	

Test Name: G000005	Class Name: Input/Output
CPU Time: 60.79 MICROSECONDS	plus or minus 3.039
Wall/CPU: 1.02 ratio.	Iteration Count: 40960
Test Description: TEXT_IO.GET an integer from a local string, timing measurement use TEXT_IO.PUT to convert 1..100 to a string, then use TEXT_IO.GET to get the number back.	

Test Name: G000006	Class Name: Input/Output
CPU Time: 177.25 MICROSECONDS	plus or minus 5.513
Wall/CPU: 1.40 ratio.	Iteration Count: 20480
Test Description: TEXT_IO.GET getting a floating point fraction from a local string, timing measurement on 0.001 to 0.01 range of numbers compare, approximately, to G000005 for integer vs float.	

Test Name:	G000007	Class Name:	Input/Output
CPU Time:	20999.91 MICROSECONDS	plus or minus	1051.536
Wall/CPU:	4.18 ratio.	Iteration Count:	320
Test Description:	Open and close an existing file, time measurement. A scratch file is created and closed. The scratch file is opened IN_FILE and closed in a loop		

Test Name: H000001	Class Name: Chapter 13
CPU Time: 1.77 MICROSECONDS	plus or minus 0.098
Wall/CPU: 1.24 ratio.	Iteration Count: 819200
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are PACKED with the pragma 'PACK.' For this test the operations are performed on the entire arrays.	

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: H000002 **Class Name:** Chapter 13
CPU Time: 42.68 MICROSECONDS plus or minus 2.134
Wall/CPU: 1.13 ratio. Iteration Count: 102400
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are NOT PACKED with the pragma 'PACK.' For this test the operations are performed on the entire array.

Test Name: H000003 **Class Name:** Chapter 13
CPU Time: 132.23 MICROSECONDS plus or minus 6.611
Wall/CPU: 1.07 ratio. Iteration Count: 51200
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are PACKED with the pragma 'PACK.' For this test the operations are performed on components in a loop.

Test Name: H000004 **Class Name:** Chapter 13
CPU Time: 44.53 MICROSECONDS plus or minus 2.227
Wall/CPU: 1.02 ratio. Iteration Count: 102400
Test Description: Time to perform standard boolean operations on arrays of booleans. For this test the arrays are NOT PACKED with the pragma 'PACK.' For this test the operations are performed on components in a loop.

Test Name: H000005 **Class Name:** Chapter 13
CPU Time: 0.13 MICROSECONDS plus or minus 0.098
Wall/CPU: 1.03 ratio. Iteration Count: 819200
Test Description: The time for UNCHECKED_CONVERSION to move one INTEGER object to another INTEGER object. This may be zero with good optimization.

Test Name: H000006 **Class Name:** Chapter 13
CPU Time: 7.45 MICROSECONDS plus or minus 0.372
Wall/CPU: 1.03 ratio. Iteration Count: 409600
Test Description: The time for UNCHECKED_CONVERSION to move 10 floating array objects to a 10 component floating record.

Test Name: H000007 **Class Name:** Chapter 13
CPU Time: 10.74 MICROSECONDS plus or minus 0.537
Wall/CPU: 1.02 ratio. Iteration Count: 409600
Test Description: The time to store and extract bit fields that are defined by representation clauses using Boolean and Integer record components. 12 accesses, 5 stores, 1 record copy.

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: H000008	Class Name: Chapter 13
CPU Time: 4.05 MICROSECONDS	plus or minus 0.203
Wall/CPU: 1.03 ratio.	Iteration Count: 409600
Test Description: The time to store and extract bit fields that are defined by nested representation clauses using packed arrays of Boolean and Integer record components.	
Test Name: H000009	Class Name: Chapter 13
CPU Time: 12.79 MICROSECONDS	plus or minus 0.640
Wall/CPU: 1.04 ratio.	Iteration Count: 409600
Test Description: The time to perform a change of representation. If the result is near zero, feature probably not implemented.	
Test Name: L000001	Class Name: Iteration
CPU Time: 0.27 MICROSECONDS	plus or minus 0.063
Wall/CPU: 1.04 ratio.	Iteration Count: 1280000
Test Description: Simple "for" loop time for I in 1 .. 100 loop time reported is for once through loop.	
Test Name: L000002	Class Name: Iteration
CPU Time: 0.27 MICROSECONDS	plus or minus 0.063
Wall/CPU: 1.04 ratio.	Iteration Count: 1280000
Test Description: Simple "while" loop time while I <= 100 loop time reported is for once through loop.	
Test Name: L000003	Class Name: Iteration
CPU Time: 0.26 MICROSECONDS	plus or minus 0.063
Wall/CPU: 1.04 ratio.	Iteration Count: 1280000
Test Description: Simple "exit" loop time loop I:=I+1; exit when I>100; end loop; time reported is for once through loop.	
Test Name: L000004	Class Name: Iteration
CPU Time: 0.30 MICROSECONDS	plus or minus 0.079
Wall/CPU: 1.04 ratio	Iteration Count: 1024000
Test Description: Measures Compiler's choice to UNWRAP a small loop of 5 iterations when given a PRAGMA OPTIMIZE(TIME). An execution time less than .05 microseconds indicates the unwrap occurred.	
Test Name: L000005	Class Name: Iteration
CPU Time: 0.21 MICROSECONDS	plus or minus 0.079
Wall/CPU: 1.03 ratio.	Iteration Count: 1024000
Test Description: Measures Compiler's choice to UNWRAP a small loop of 5 iterations when given a PRAGMA OPTIMIZE(Space). An execution speed < .05 microseconds indicates the unwrap occurred.	

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: P000001 Class Name: Procedure
 CPU Time: 0.06 MICROSECONDS plus or minus 0.079
 Wall/CPU: 1.05 ratio. Iteration Count: 1024000
 Test Description: Procedure call and return time (may be zero if automatic inlining) procedure is local, no parameters.

Test Name: P000002 Class Name: Procedure
 CPU Time: 7.15 MICROSECONDS plus or minus 0.357
 Wall/CPU: 1.05 ratio. Iteration Count: 512000
 Test Description: Procedure call and return time. Procedure is local, no parameters when procedure is not inlinable.

Test Name: P000003 Class Name: Procedure
 CPU Time: 7.32 MICROSECONDS plus or minus 0.366
 Wall/CPU: 1.02 ratio. Iteration Count: 512000
 Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. Compare to P000002.

Test Name: P000004 Class Name: Procedure
 CPU Time: 0.48 MICROSECONDS plus or minus 0.079
 Wall/CPU: 1.04 ratio. Iteration Count: 1024000
 Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package pragma INLINE used. Compare to P000001.

Test Name: P000005 Class Name: Procedure
 CPU Time: 7.95 MICROSECONDS plus or minus 0.397
 Wall/CPU: 1.04 ratio. Iteration Count: 512000
 Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. One parameter, in INTEGER

Test Name: P000006 Class Name: Procedure
 CPU Time: 8.16 MICROSECONDS plus or minus 0.408
 Wall/CPU: 1.03 ratio. Iteration Count: 512000
 Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. One parameter, out INTEGER.

Test Name: P000007 Class Name: Procedure
 CPU Time: 8.44 MICROSECONDS plus or minus 0.422
 Wall/CPU: 1.04 ratio. Iteration Count: 512000
 Test Description: Procedure call and return time measurement. The procedure is in a separately compiled package. One parameter, in out INTEGER

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: P000010 Class Name: Procedure
CPU Time: 15.43 MICROSECONDS plus or minus 0.771
Wall/CPU: 1.06 ratio. Iteration Count: 256000
Test Description: Procedure call and return time measurement. Compare to P000005 10 parameters, in INTEGER

Test Name: P000011 Class Name: Procedure
CPU Time: 22.89 MICROSECONDS plus or minus 1.144
Wall/CPU: 1.04 ratio. Iteration Count: 256000
Test Description: Procedure call and return time measurement. Compare to P000005, P000010 20 parameters, in INTEGER.

Test Name: P000012 Class Name: Procedure
CPU Time: 10.59 MICROSECONDS plus or minus 0.529
Wall/CPU: 1.02 ratio. Iteration Count: 256000
Test Description: Procedure call and return time measurement. Compare with P000010 (discrete vs composite parameters) 10 parameters, in MY_RECORD a 3 component record.

Test Name: P000013 Class Name: Procedure
CPU Time: 14.22 MICROSECONDS plus or minus 0.711
Wall/CPU: 1.04 ratio. Iteration Count: 256000
Test Description: Procedure call and return time measurement 20 composite 'in' parameters. The composite type is a 3 component record.

Test Name: T000001 Class Name: Tasking
CPU Time: 176.95 MICROSECONDS plus or minus 8.848
Wall/CPU: 1.05 ratio. Iteration Count: 25600
Test Description: Minimum rendezvous, entry call and return time 1 task 1 entry, task inside procedure no select.

Test Name: T000002 Class Name: Tasking
CPU Time: 175.00 MICROSECONDS plus or minus 8.750
Wall/CPU: 1.06 ratio. Iteration Count: 25600
Test Description: Task entry call and return time measured. One task active, 1 entry in task, task in a package, no select statement.

Test Name: T000003 Class Name: Tasking
CPU Time: 174.22 MICROSECONDS plus or minus 8.711
Wall/CPU: 1.03 ratio. Iteration Count: 25600
Test Description: Task entry call and return time measured. Two tasks active, 1 entry per task, tasks in a package, no select statement.

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Test Name: T000004 **Class Name:** Tasking
CPU Time: 217.19 MICROSECONDS **plus or minus** 10.859
Wall/CPU: 1.04 ratio. **Iteration Count:** 25600
Test Description: Task entry call and return time measured. One tasks active, 2 entries, tasks in a package, using select statement.

Test Name: T000005 **Class Name:** Tasking
CPU Time: 226.25 MICROSECONDS **plus or minus** 11.312
Wall/CPU: 1.01 ratio. **Iteration Count:** 32000
Test Description: Task entry call and return time measured. Ten tasks active, 1 entry per task, tasks in a package, no select statement.

Test Name: T000006 **Class Name:** Tasking
CPU Time: 304.38 MICROSECONDS **plus or minus** 15.219
Wall/CPU: 1.01 ratio. **Iteration Count:** 16000
Test Description: Task entry call and return time measurement. One task with 10 entries , task in a package, 1 select statement, compare to T000005.

Test Name: T000008 **Class Name:** Tasking
CPU Time: 388.28 MICROSECONDS **plus or minus** 19.414
Wall/CPU: 1.01 ratio. **Iteration Count:** 12800
Test Description: Measure the average time to pass an integer from a producer task through a buffer task to a consumer task.

Test Name: Y000001 Measure actual delay vs commanded delay

Commanded	Actual	CPU	Iterations
0.0010	0.0100	0.0000	1024
0.0020	0.0099	0.0000	512
0.0039	0.0099	0.0000	256
0.0078	0.0099	0.0000	128
0.0156	0.0200	0.0000	64
0.0313	0.0400	0.0000	32
0.0625	0.0699	0.0000	16
0.1250	0.1299	0.0000	8
0.2500	0.2500	0.0000	4
0.5000	0.5000	0.0000	2
1.0000	1.0000	0.0000	2
2.0000	2.0000	0.0000	2

\$ RUN A000055
%ADA-F-NAME_ERROR, NAME_ERROR
-RMS-E-FNF, file not found
-ADA-I-OPERINFO, While performing TEXT_IO.OPEN
-ADA-I-MODEINFO, The Mode of the file is IN_FILE

APPENDIX I
PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

-ADA-I-TEXTINFO, Column 1, Line 1, Page 1, Line length is 0, Page length is 0
 -ADA-I-FILENAME, The external file specification is:
 -ADA-I-MOREINFO, \$1\$DIA9:[DAVIDSON.PIWG]A000052D.;
 %TRACE-E-TRACEBACK, symbolic stack dump follows
 module name routine name line rel P
 abs PC

000553AD

000553AD

----- above condition handler called with exception 0031A264:
 %ADA-F-NAME_ERROR, NAME_ERROR
 -RMS-E-FNF, file not found
 -ADA-I-OPERINFO, While performing TEXT_IO.OPEN
 -ADA-I-MODEINFO, The Mode of the file is IN_FILE
 -ADA-I-TEXTINFO, Column 1, Line 1, Page 1, Line length is 0, Page length is 0
 -ADA-I-FILENAME, The external file specification is:
 -ADA-I-MOREINFO, \$1\$DIA9:[DAVIDSON.PIWG]A000052D.;
 ----- end of exception message

0004A152 0004A152

0004FB57 0004FB57

0004E11D 0004E11D
 A000055 A000055 69 00000131

00003774
 ADA\$ELAB_A00005 ADA\$ELAB_A000055 00000009

00002409 000043C9

000043C9 0005517D

0005517D 0000001B
 ADA\$ELAB_A00005 ADA\$ELAB_A000055

0000241B 000043A4

000043A4
 \$ DEL SCRATCH*.*;* !remove scratch files from "G" tests
 \$ DEL A000052D.*;* !remove intermediate timing files
 %DELETE-W-SEARCHFAIL, error searching for \$1\$DIA9:[DAVIDSON.PIWG]A000052D.*
 -RMS-E-FNF, file not found
 \$ DEL PIWGRES.*;* !remove saved results if any
 %DELETE-W-SEARCHFAIL, error searching for \$1\$DIA9:[DAVIDSON.PIWG]PIWGRES.*
 -RMS-E-FNF, file not found
 \$!DEL [.ADALIB]*.*;* !remove compilation library
 DAVIDSON job terminated at 11-JUN-1993 18:44:37.78

APPENDIX I

PIWG (FIRST) EXECUTION TEST RESULTS USING DEC ADA ON THE VAX COMPUTER

Accounting information:

Buffered I/O count:	15713	Peak working set size:	3862
Direct I/O count:	24392	Peak page file size:	11000
Page faults:	393869	Mounted volumes:	0
Charged CPU time:	0 00:47:03.21	Elapsed time:	0 01:25:08.64

APPENDIX J

**COMMAND FILE FOR
"SECOND RUN" PIWG TESTS
ON THE VAX COMPUTER**

APPENDIX J
COMMAND FILE FOR "SECOND RUN" PIWG TESTS ON THE VAX COMPUTER

```
$! FOR PIWG ON THE VAX VMS
$! THIS IS FOR "SECOND RUN" AS DEFINED IN THE
$! "READ.ME" FILE FOR PIWG
$!
$! VAX VMS files to compile (using standard output PIWG_IO)
$ SET VERIFY
$ SET NOON
$ SET DEF $!$DIA9:[DAVIDSON.PIWG]
$ DEL [.ADALIB]*.*;*    ! Clean out any old libraries
$ ACS CREATE LIB [.ADALIB]
$ ACS SET LIB [.ADALIB]
$ @ VAXCONFIG
$!
$ ADA A000001 ! DURATION_IO instantiation
$ ADA A000012 ! CPU_TIME_CLOCK.VAX
$ ADA A000051 ! wall timing routines for host only
$ ACS LINK A000051
$!
$ RUN A000051
$ RUN A000051 ! calibrate time to measure time
$ RUN A000051
$ ADA Z000001
$ ADA Z000002
$ ADA Z000003
$ ADA Z000004
$ ADA Z000005
$ ADA Z000006
$ ADA Z000007
$ ADA Z000008
$ ADA Z000009
$ ADA Z000010
$ ADA Z000011
$ ADA Z000012
$ ADA Z000013
$ ADA Z000014
$ ADA Z000015
$ ADA Z000016
$ ADA Z000016A
$ ADA Z000017
$ ADA Z000017A
$ ADA Z000018
$ ACS LINK Z000018
$ RUN Z000018
$!
$ ADA Z000020
$ ADA Z000021
```

APPENDIX J
COMMAND FILE FOR "SECOND RUN" PIWG TESTS ON THE VAX COMPUTER

```
$ ADA Z000022
$ ADA Z000023
$ ACS LINK Z000023 $ RUN Z000023
$ RUN A000051 ! final time measurement
$!           when this test can be run with no other tasks running, it represents a
$!           composite software development benchmark.
$!

$ DEL SCRATCH*.*;* ! remove scratch files from "G" tests
$ DEL A000052D.* ! remove intermediate timing files
$ DEL PIWGRES.* ! remove saved results if any
$! DEL [.ADALIB]*.*;* ! remove compilation library
```

APPENDIX K

**PIWG "SECOND RUN" RESULTS
USING
DEC ADA ON THE VAX COMPUTER**

APPENDIX K
PIWG "SECOND RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$!SYLOGIN.COM
$! For the AAAF CLuster
$ set NOVERIFY
$ SET NOON
$ SET DEF $!$DIA9:[DAVIDSON.PIWG]
$ DEL [.ADALIB]*.*;* ! Clean out any old libraries
$ ACS CREATE LIB [.ADALIB]
%ACS-I-CL_LIBCRE, Library $!$DIA9:[DAVIDSON.PIWG.ADALIB] created
$ ACS SET LIB [.ADALIB]
%ACS-I-CL_LIBIS, Current program library is
$!$DIA9:[DAVIDSON.PIWG.ADALIB]
$ @ VAXCONFIG
    System configuration:
        Cluster:  AAAF
        Node:     JEDI
        CPU:      VAX 4000-200
            Character emulated:  TRUE
            F_FLOAT emulated:    FALSE
            D_FLOAT emulated:    FALSE
            G_FLOAT emulated:    FALSE
            H_FLOAT emulated:    TRUE
        VMS version:  V5.5
        Virtual page count: 230000
        Working set maximum: 16400
        Main Memory (32.00Mb)
        System device: RF72
        User device:  RF71
    VAX Ada software configuration:
        ADA version: "VAX Ada V2.2-38"
        ACS version: "VAX Ada V2.2-38"
        ADARTL version: "V5.4-03"
        ADAMSG version: "1-029"
    Process configuration:
        Open file limit:      50
        Enqueue quota:       100
        Timer queue quota:   20
        Page file quota:     17000
        Working set memory parameters:
            WSQUOTA: 2500
            WSEXTENT: 4000
            Adjustment: enabled
$ EXIT ! 0
$!
$ ADA A000001 ! DURATION_IO instantiation
$ ADA A000012 ! CPU_TIME_CLOCK.VAX
$ ADA A000051 ! wall timing routines for host only
$ ACS LINK A000051
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[A0000051-
SYSS$INPUT:/OPTIONS

```

APPENDIX K
PIWG "SECOND RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$!$DIA9:[DAVIDSON.PIWG]A000051.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000051.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000051.COM;1
$EXIT
$!
$ RUN A000051
    CPU time now=    22.8199    WALL time now= 41478.8800 seconds.
$ RUN A000051 ! calibrate time to measure time
    CPU time now=    22.9500    WALL time now= 41479.1300 seconds.
$ RUN A000051
    CPU time now=    23.0800    WALL time now= 41479.4200 seconds.
$ ADA Z000001
$ ADA Z000002
$ ADA Z000003
$ ADA Z000004
$ ADA Z000005
$ ADA Z000006
$ ADA Z000007
$ ADA Z000008
$ ADA Z000009
$ ADA Z000010
$ ADA Z000011
$ ADA Z000012
$ ADA Z000013
$ ADA Z000014
$ ADA Z000015
$ ADA Z000016
$ ADA Z000016A
$ ADA Z000017
$ ADA Z000017A
$ ADA Z000018
$ ACS LINK Z000018
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK:= ""
$LINK-
/NOMAP-
/EXE=[Z000018-
    SY$INPUT:/OPTIONS
    $!$DIA9:[DAVIDSON.PIWG]Z000018.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]Z000018.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]Z000018.COM;1
$EXIT
$ RUN Z000018
    Test printout and value of acceleration, 9.80665E+00 meter per second squared = G
    1.10325E+01 meter
    1.50000E+00 second
    2.08030E+01 meter per second
$!
$ ADA Z000020
$ ADA Z000021
$ ADA Z000022

```

APPENDIX K
PIWG "SECOND RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$ ADA Z000023
$ ACS LINK Z000023
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE-[]Z000023-
      SY$INPUT:/OPTIONS
      $!$DIA9:[DAVIDSON.PIWG]Z000023.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]Z000023.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]Z000023.COM;1
$EXIT
$ RUN Z000023
  UP SORTED DATA
    1 1.00000E+00 AAA FIRST      1.09
    2 2.00000E+00 BBB SECOND    2.09
    3 3.00000E+00 CCC THIRD     3.09
    4 4.00000E+00 DDD FOURTH    4.09
  DOWN SORTED DATA
    4 4.00000E+00 DDD FOURTH    4.09
    3 3.00000E+00 CCC THIRD     3.09
    2 2.00000E+00 BBB SECOND    2.09
    1 1.00000E+00 AAA FIRST     1.09
in the bag
gone fishing
end FISH
ALL_STATEMENTS_PROCEDURE_2
into LOOP_NAME_1
Z000021 finished
$ RUN A000051 ! final time measurement
CPU time now= 695.4900    WALL time now= 42445.5900 seconds.
$!      when this test can be run with no other tasks running, it represents a composite
$!      development benchmark.
$!

$ DEL SCRATCH*.*;* ! remove scratch files from "G" tests
%DELETE-W-SEARCHFAIL, error searching for
$!$DIA9:[DAVIDSON.PIWG]SCRATCH*.*;*
-RMS-E-FNF, file not found
$ DEL A000052D.* ! remove intermediate timing files
%DELETE-W-SEARCHFAIL, error searching for $!$DIA9:[DAVIDSON.PIWG]A000052D.*
-RMS-E-FNF, file not found
$ DEL PIWGRES.* ! remove saved results if any
%DELETE-W-SEARCHFAIL, error searching for $!$DIA9:[DAVIDSON.PIWG]PIWGRES.*
-RMS-E-FNF, file not found
$! DEL [.ADALIB]*.*;* ! remove compilation library
      DAVIDSON    job terminated at 16-JUN-1993 11:47:26.54

```

APPENDIX K
PIWG "SECOND RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

Accounting information:

Buffered I/O count: 2495
Direct I/O count: 4085
Page faults: 120251
Charged CPU time: 0 00:11:35.94

Peak working set size: 4000
Peak page file size: 11527
Mounted volumes: 0
Elapsed time: 0 00:17:39.19

APPENDIX L

**COMMAND FILE FOR
"THIRD RUN" PIWG TESTS
ON THE VAX COMPUTER**

APPENDIX L
COMMAND FILE FOR "THIRD RUN" PIWG TESTS ON THE VAX COMPUTER

```
$! FOR PIWG ON THE VAX VMS
$! THIS IS FOR "THIRD RUN" AS DEFINED IN THE
$! "READ.ME" FILE FOR PIWG
$!
$! VAX VMS files to compile (using standard output PIWG_IO)
$ SET VERIFY
$ SET NOON
$ SET DEF $!$DIA9:[DAVIDSON.PIWG]
$ DEL [.ADALIB]*.*;* ! Clean out any old libraries
$ ACS CREATE LIB [.ADALIB]
$ ACS SET LIB [.ADALIB]
$ @ VAXCONFIG
$!
$ ADA A000001 ! DURATION_IO instantiation
$ ADA A000012 ! CPU_TIME_CLOCK.VAX
$ ADA A000051 ! wall timing routines for host only
$ ACS LINK A000051
$!
$ RUN A000051
$ RUN A000051 ! calibrate time to measure time
$ RUN A000051
$ ADA Z000100
$ ADA Z000101
$ ADA Z000102
$ ADA Z000103
$ ADA Z000104
$ ADA Z000105
$ ADA Z000106
$ ADA Z000107
$ ADA Z000108
$ ADA Z000109
$ ADA Z000110
$ ADA Z000111
$ ADA Z000112
$ ADA Z000113
$ ADA Z000114
$ ADA Z000115
$ ADA Z000116
$ ADA Z000117
$ ADA Z000118
$ ADA Z000119
$ ADA Z000120
$ ADA Z000121
$ ADA Z000122
$ ADA Z000123
$ ADA Z000124
$ ADA Z000125
$ ADA Z000126
$ ADA Z000127
$ ADA Z000128
$ ADA Z000129
$ ADA Z000130
$ ADA Z000131
$ ADA Z000132
```

APPENDIX L
COMMAND FILE FOR "THIRD RUN" PIWG TESTS ON THE VAX COMPUTER

\$ ADA Z000133
\$ ADA Z000134
\$ ADA Z000135
\$ ADA Z000136
\$ ADA Z000137
\$ ADA Z000138
\$ ADA Z000139
\$ ADA Z000140
\$ ADA Z000141
\$ ADA Z000142
\$ ADA Z000143
\$ ADA Z000144
\$ ADA Z000145
\$ ADA Z000146
\$ ADA Z000147
\$ ADA Z000148
\$ ADA Z000149
\$ ADA Z000150
\$ ADA Z000151
\$ ADA Z000152
\$ ADA Z000153
\$ ADA Z000154
\$ ADA Z000155
\$ ADA Z000156
\$ ADA Z000157
\$ ADA Z000158
\$ ADA Z000159
\$ ADA Z000160
\$ ADA Z000161
\$ ADA Z000162
\$ ADA Z000163
\$ ADA Z000164
\$ ADA Z000165
\$ ADA Z000166
\$ ADA Z000167
\$ ADA Z000168
\$ ADA Z000169
\$ ADA Z000170
\$ ADA Z000171
\$ ADA Z000172
\$ ADA Z000173
\$ ADA Z000174
\$ ADA Z000175
\$ ADA Z000176
\$ ADA Z000177
\$ ADA Z000178
\$ ADA Z000179
\$ ADA Z000180
\$ ADA Z000181
\$ ADA Z000182
\$ ADA Z000183
\$ ADA Z000184
\$ ADA Z000185
\$ ADA Z000186

APPENDIX L
COMMAND FILE FOR "THIRD RUN" PIWG TESTS ON THE VAX COMPUTER

```
$ ADA Z000187
$ ADA Z000188
$ ADA Z000189
$ ADA Z000190
$ ADA Z000191
$ ADA Z000192
$ ADA Z000193
$ ADA Z000194
$ ADA Z000195
$ ADA Z000196
$ ADA Z000197
$ ADA Z000198
$ ADA Z000199
$ RUN A000051 ! final wall time
$!
$!
$ RUN A000051 ! initial wall time
$ ADA Z000200
$ RUN A000051 ! final time measurement
$!          when this test can be run with no other tasks running, it represents a
$!          composite software development benchmark. Compute difference and
$!          report.
$!
$ DEL SCRATCH*.*;* ! remove scratch files from "G" tests
$ DEL A000052D.*;* ! remove intermediate timing files
$ DEL PIWGRES.*;* ! remove saved results if any
$! DEL [.ADALIB]*.*;* ! remove compilation library
```


APPENDIX M

**PIWG "THIRD RUN" RESULTS
USING
DEC ADA ON THE VAX COMPUTER**

APPENDIX M
PIWG "THIRD RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

```
$!SYLOGIN.COM
$! For the AAAF CLuster
$ set NOVERIFY
$ SET NOON
$ SET DEF $!$DIA9:[DAVIDSON.PIWG]
$ DEL [.ADALIB]*.*;* ! Clean out any old libraries
$ ACS CREATE LIB [.ADALIB]
%ACS-I-CL_LIBCRE, Library $!$DIA9:[DAVIDSON.PIWG.ADALIB] created
$ ACS SET LIB [.ADALIB]
%ACS-I-CL_LIBIS, Current program library is
$!$DIA9:[DAVIDSON.PIWG.ADALIB]
$ @ VAXCONFIG
  System configuration:
    Cluster:  AAAF
    Node:      JEDI
    CPU:       VAX 4000-200
      Character emulated:  TRUE
      F_FLOAT emulated:   FALSE
      D_FLOAT emulated:   FALSE
      G_FLOAT emulated:   FALSE
      H_FLOAT emulated:   TRUE
    VMS version: V5.5
    Virtual page count: 230000
    Working set maximum: 16400
    Main Memory (32.00Mb)
    System device: RF72
    User device:  RF71
  VAX Ada software configuration:
    ADA version: "VAX Ada V2.2-38"
    ACS version: "VAX Ada V2.2-38"
    ADARTL version: "V5.4-03"
    ADAMSG version: "1-029"
  Process configuration:
    Open file limit:      50
    Enqueue quota:        100
    Timer queue quota:    20
    Page file quota:      17000
    Working set memory parameters:
      WSQUOTA: 2500
      WSEXTENT: 4000
      Adjustment: enabled
$ EXIT
$!
$ ADA A000001 ! DURATION_IO instantiation
$ ADA A000012 ! CPU_TIME_CLOCK.VAX
$ ADA A000051 ! wall timing routines for host only
$ ACS LINK A000051
%ACS-I-CL_LINKING, Invoking the VMS Linker for VAX_VMS target
$SET DEFAULT $!$DIA9:[DAVIDSON.PIWG]
$LINK := ""
$LINK-
/NOMAP-
/EXE=[]A000051-
SYSSINPUT:/OPTIONS
```

APPENDIX M
PIWG "THIRD RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$!$DIA9:[DAVIDSON.PIWG]A000051.OBJ;1
$STATUS = $STATUS
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000051.OBJ;1
$DELETE $!$DIA9:[DAVIDSON.PIWG]A000051.COM;1
$EXIT
$!
$ RUN A000051
    CPU time now= 33.5699    WALL time now= 48105.2600 seconds.
$ RUN A000051 ! calibrate time to measure time
    CPU time now= 33.6700    WALL time now= 48105.4900 seconds.
$ RUN A000051
    CPU time now= 33.7800    WALL time now= 48105.7700 seconds.
$ ADA Z000100
$ ADA Z000101
$ ADA Z000102
$ ADA Z000103
$ ADA Z000104
$ ADA Z000105
$ ADA Z000106
$ ADA Z000107
$ ADA Z000108
$ ADA Z000109
$ ADA Z000110
$ ADA Z000111
$ ADA Z000112
$ ADA Z000113
$ ADA Z000114
$ ADA Z000115
$ ADA Z000116
$ ADA Z000117
$ ADA Z000118
$ ADA Z000119
$ ADA Z000120
$ ADA Z000121
$ ADA Z000122
$ ADA Z000123
$ ADA Z000124
$ ADA Z000125
$ ADA Z000126
$ ADA Z000127
$ ADA Z000128
qqqqqqqr$ ADA Z000129
$ ADA Z000130
$ ADA Z000131
$ ADA Z000132
$ ADA Z000133
$ ADA Z000134
$ ADA Z000135
$ ADA Z000136
$ ADA Z000137
$ ADA Z000138
$ ADA Z000139
$ ADA Z000140
$ ADA Z000141

```

APPENDIX M
PIWG "THIRD RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

\$ ADA Z000142
\$ ADA Z000143
\$ ADA Z000144
\$ ADA Z000145
\$ ADA Z000146
\$ ADA Z000147
\$ ADA Z000148
\$ ADA Z000149
\$ ADA Z000150
\$ ADA Z000151
\$ ADA Z000152
\$ ADA Z000153
\$ ADA Z000154
\$ ADA Z000155
\$ ADA Z000156
\$ ADA Z000157
\$ ADA Z000158
\$ ADA Z000159
\$ ADA Z000160
\$ ADA Z000161
\$ ADA Z000162
\$ ADA Z000163
\$ ADA Z000164
\$ ADA Z000165
\$ ADA Z000166
\$ ADA Z000167
\$ ADA Z000168
\$ ADA Z000169
\$ ADA Z000170
\$ ADA Z000171
\$ ADA Z000172
\$ ADA Z000173
\$ ADA Z000174
\$ ADA Z000175
\$ ADA Z000176
\$ ADA Z000177
\$ ADA Z000178
\$ ADA Z000179
\$ ADA Z000180
\$ ADA Z000181
\$ ADA Z000182
\$ ADA Z000183
\$ ADA Z000184
\$ ADA Z000185
\$ ADA Z000186
\$ ADA Z000187
\$ ADA Z000188
\$ ADA Z000189
\$ ADA Z000190
\$ ADA Z000191
\$ ADA Z000192
\$ ADA Z000193
\$ ADA Z000194
\$ ADA Z000195

APPENDIX M
PIWG "THIRD RUN" RESULTS USING DEC ADA ON THE VAX COMPUTER

```

$ ADA Z000196
$ ADA Z000197
$ ADA Z000198
$ ADA Z000199
$ RUN A000051 ! final wall time
    CPU time now= 307.7400    WALL time now= 48836.6800 seconds.
$!
$!
$ RUN A000051 ! initial wall time
    CPU time now= 307.8500    WALL time now= 48836.9500 seconds.
$ ADA Z000200
$ RUN A000051 ! final time measurement
    CPU time now= 536.2300    WALL time now= 49496.1500 seconds.
$!      when this test can be run with no other tasks running, it represents a composite
$!      software development benchmark. Compute difference and report.
$!
$ DEL SCRATCH*.*;* ! remove scratch files from "G" tests
%DELETE-W-SEARCHFAIL, error searching for
$!$DIA9:[DAVIDSON.PIWG]SCRATCH*.*;*
-RMS-E-FNF, file not found
$ DEL A000052D.*;* ! remove intermediate timing files
%DELETE-W-SEARCHFAIL, error searching for $!$DIA9:[DAVIDSON.PIWG]A000052D.*;*
-RMS-E-FNF, file not found
$ DEL PIWGRES.*;* ! remove saved results if any
%DELETE-W-SEARCHFAIL, error searching for $!$DIA9:[DAVIDSON.PIWG]PIWGRES.*;*
-RMS-E-FNF, file not found
$! DEL [.ADALIB]*.*;* ! remove compilation library
    DAVIDSON    job terminated at 16-JUN-1993 13:44:57.17

```

Accounting information:

Buffered I/O count:	11510	Peak working set size:	3999
Direct I/O count:	20524	Peak page file size:	11052
Page faults:	368280	Mounted volumes:	0
Charged CPU time:	0 00:08:56.70	Elapsed time:	0 00:28:22.82